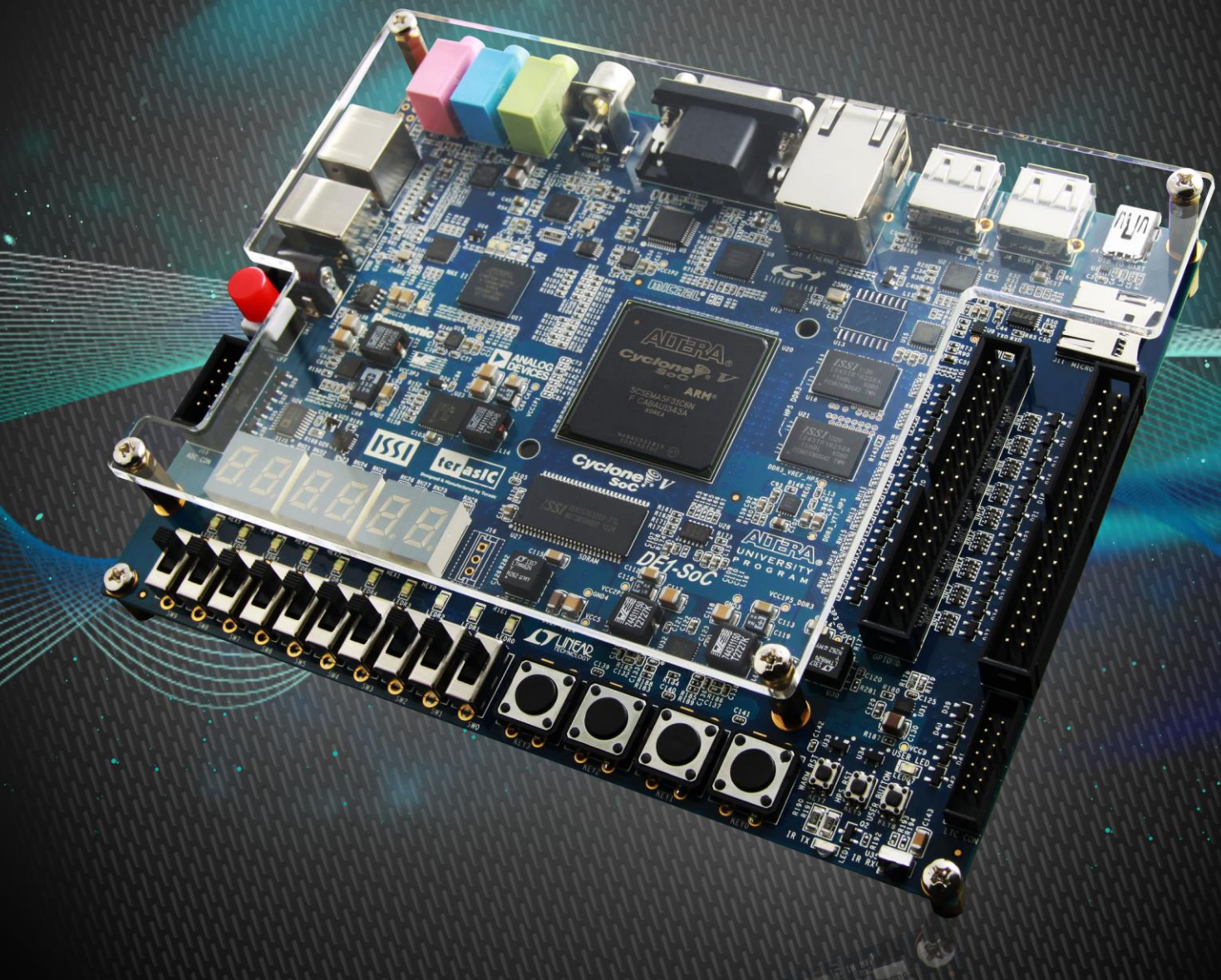
DE1-SoC

User Manual



terasIC
www.terasic.com

ALTERA
UNIVERSITY
PROGRAM

Copyright © 2003-2013 Terasic Technologies Inc. All Rights Reserved.

CONTENTS

CHAPTER 1 DE1-SOC DEVELOPMENT KIT 3

1.1 PACKAGE CONTENTS 3
1.2 DE1-SOC SYSTEM CD 4
1.3 GETTING HELP 4

CHAPTER 2 INTRODUCTION OF THE DE1-SOC BOARD 5

2.1 LAYOUT AND COMPONENTS 5
2.2 BLOCK DIAGRAM OF THE DE1-SOC BOARD 7

CHAPTER 3 USING THE DE1-SOC BOARD 10

3.1 FPGA CONFIGURATION MODE SETTING 10
3.2 CONFIGURING THE CYCLONE V SOC FPGA 11
3.3 BOARD STATUS ELEMENTS 17
3.4 BOARD RESET ELEMENTS 17
3.5 CLOCK CIRCUITRY 19
3.6 INTERFACE ON FPGA 20
3.6.1 USER PUSH-BUTTONS, SWITCHES AND LEDS ON FPGA 21
3.6.2 USING THE 7-SEGMENT DISPLAYS 24
3.6.3 USING THE 2X20 GPIO EXPANSION HEADERS 26
3.6.4 USING THE 24-BIT AUDIO CODEC 28
3.6.5 I2C MULTIPLEXER 29
3.6.6 VGA 30
3.6.7 TV DECODER 33
3.6.8 IR RECEIVER 35
3.6.9 IR EMITTER LED 35
3.6.10 SDRAM MEMORY ON FPGA 36
3.6.11 PS/2 SERIAL PORT 38
3.6.12 A/D CONVERTER AND 2X5 HEADER 40
3.7 INTERFACE ON HARD PROCESSOR SYSTEM (HPS) 41
3.7.1 USER PUSH-BUTTON AND LED ON HPS 41
3.7.2 GIGABIT ETHERNET 42
3.7.3 UART 43
3.7.4 DDR3 MEMORY ON HPS 44
3.7.5 MICRO SD 46
3.7.6 2-PORT USB HOST 47
3.7.7 G-SENSOR 48

3.7.8 LTC CONNECTOR 49

CHAPTER 4 DE1-SOC SYSTEM BUILDER 51

4.1 INTRODUCTION 51
4.2 GENERAL DESIGN FLOW 51
4.3 USING DE1-SOC SYSTEM BUILDER 52

CHAPTER 5 EXAMPLES FOR FPGA 58

5.1 DE1-SOC FACTORY CONFIGURATION 58
5.2 AUDIO RECORDING AND PLAYING 59
5.3 A KARAOKE MACHINE 62
5.4 SDRAM TEST BY NIOS II 64
5.5 SDRAM RTL TEST 67
5.6 TV BOX DEMONSTRATION 69
5.7 PS/2 MOUSE DEMONSTRATION 71
5.8 IR EMITTER LED AND RECEIVER DEMONSTRATION 74
5.9 ADC READING 80

CHAPTER 6 EXAMPLES FOR HPS SOC 83

6.1 HELLO PROGRAM 83
6.2 USERS LED AND KEY 85
6.3 I2C INTERFACED G-SENSOR 91
6.4 I2C MUX TEST 94

CHAPTER 7 EXAMPLES FOR USING BOTH HPS SOC AND FGPA 97

7.1 HPS CONTROL LED AND HEX 97

CHAPTER 8 STEPS OF PROGRAMMING THE QUAD SERIAL CONFIGURATION DEVICE 101

8.1 BEFORE YOU BEGIN 101
8.2 CONVERT. SOF FILE TO .JIC FILE 101
8.3 WRITE JIC FILE INTO QUAD SERIAL CONFIGURATION DEVICE 106
8.4 ERASE THE QUAD SERIAL CONFIGURATION DEVICE 107

CHAPTER 9 APPENDIX 109

9.1 REVISION HISTORY 109
9.2 COPYRIGHT STATEMENT 109

Chapter 1

DE1-SoC Development Kit

The DE1-SoC Development Kit presents a robust hardware design platform built around the Altera System-on-Chip (SoC) FPGA, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility. Users can now leverage the power of tremendous re-configurability paired with a high-performance, low-power processor system. Altera's SoC integrates an ARM-based hard processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone. The DE1-SoC development board includes hardware such as high-speed DDR3 memory, video and audio capabilities, Ethernet networking, and much more.

The DE1-SoC Development Kit contains all components needed to use the board in conjunction with a computer that runs the Microsoft Windows XP or later.

1.1 Package Contents

Figure 1-1 shows a photograph of the DE1-SoC package.

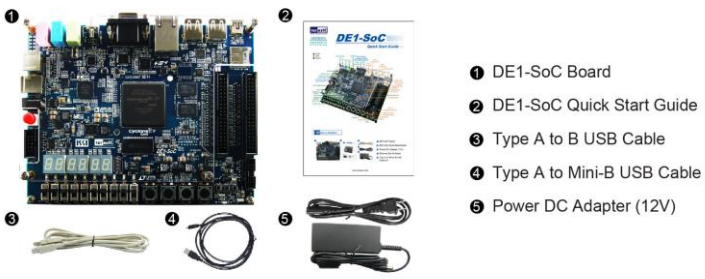


Figure 1-1 The DE1-SoC package contents

The DE1-SoC package includes:

- The DE1-SoC development board
- DE1-SoC Quick Start Guide
- USB Cable (Type A to B) for FPGA programming and control
- USB Cable (Type A to Mini-B) for UART control
- 12V DC power adapter

1.2 DE1-SoC System CD

The DE1-SoC System CD contains all DE1-SoC documentation and supporting materials, including the User Manual, System Builder, reference designs and device datasheets. User can download this System CD from the link: http://de1-soc.terasic.com.

1.3 Getting Help

Here are the addresses where you can get help if you encounter any problems:

- Altera Corporation
- 101 Innovation Drive San Jose, California, 95134 USA

Email: university@altera.com

- Terasic Technologies
- 9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan

Email: support@terasic.com

Tel.: +886-3-575-088()

Web: de1-soc.terasic.com

Chapter 2

Introduction of the DE1-SoC Board

This chapter presents the features and design characteristics of the board.

2.1 Layout and Components

A photograph of the board is shown in **Figure 2-1.** It depicts the layout of the board and indicates the location of the connectors and key components.

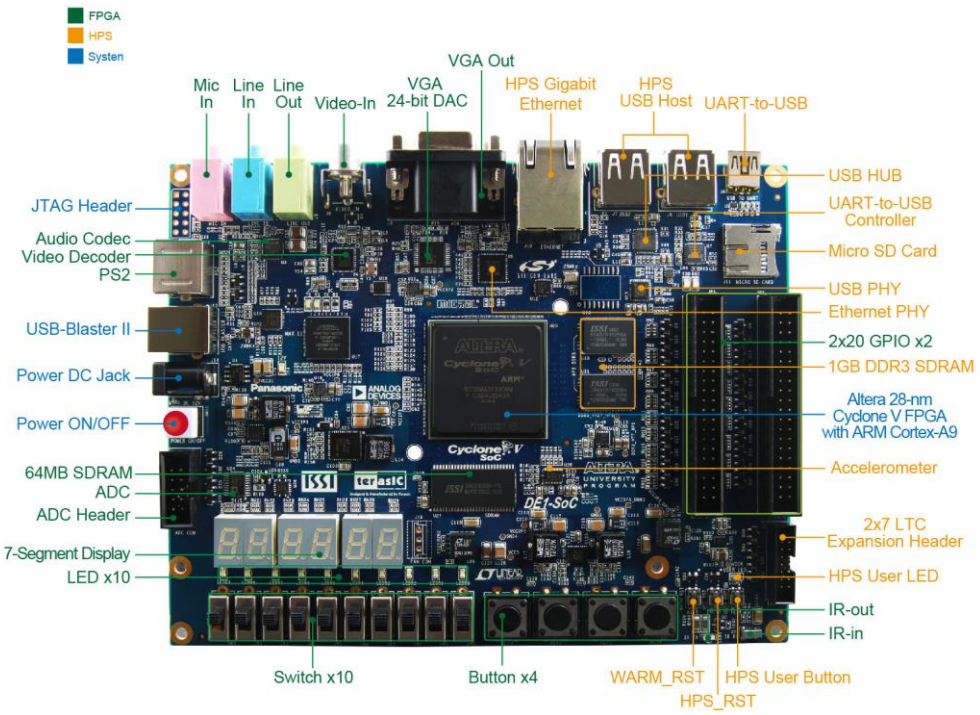


Figure 2-1 Development Board (top view)

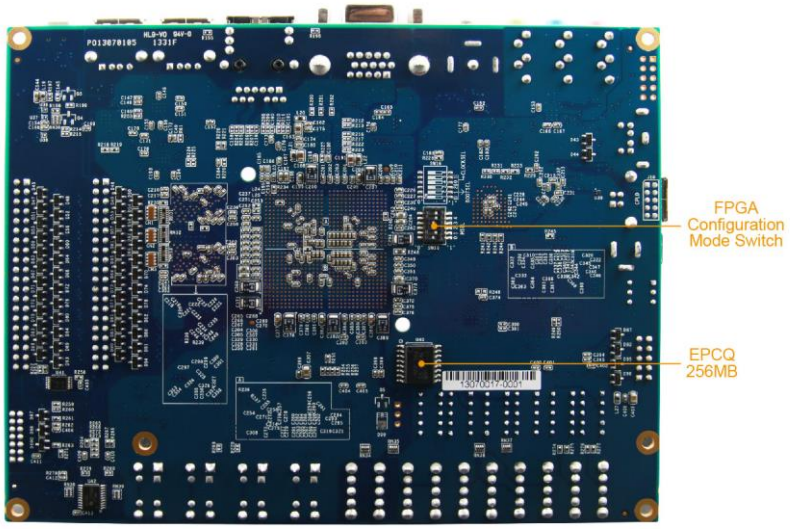


Figure 2-2 Development Board (bottom view)

The DE1-SoC board has many features that allow users to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware is provided on the board:

■ FPGA

- Altera Cyclone® V SE 5CSEMA5F31C6N device
- Altera Serial Configuration device – EPCQ256
- USB Blaster II(on board) for programming; JTAG Mode
- 64MB SDRAM (16-bit data bus)
- 4 Push-buttons
- 10 Slide switches
- 10 Red user LEDs
- Six 7-segment displays
- Four 50MHz clock sources from clock generator
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- PS/2 mouse/keyboard connector
- IR receiver and IR emitter
- Two 40-pin Expansion Header with diode protection
- A/D Converter, 4-pin SPI interface with FPGA

■ HPS (Hard Processor System)

- 800MHz Dual-core ARM Cortex-A9 MPCore processor
- 1GB DDR3 SDRAM (32-bit data bus)
- 1 Gigabit Ethernet PHY with RJ45 connector
- 2-port USB Host, Normal Type-A USB connector
- Micro SD card socket
- Accelerometer (I2C interface + interrupt)
- UART to USB, USB Mini-B connector
- Warm reset button and cold reset button
- One user button and one user LED
- LTC 2x7 expansion header

2.2 Block Diagram of the DE1-SoC Board

Figure 2-3 gives the block diagram of the board. To provide maximum flexibility for the user, all connections are made through the Cyclone V SoC FPGA device. Thus, the user can configure the FPGA to implement any system design.

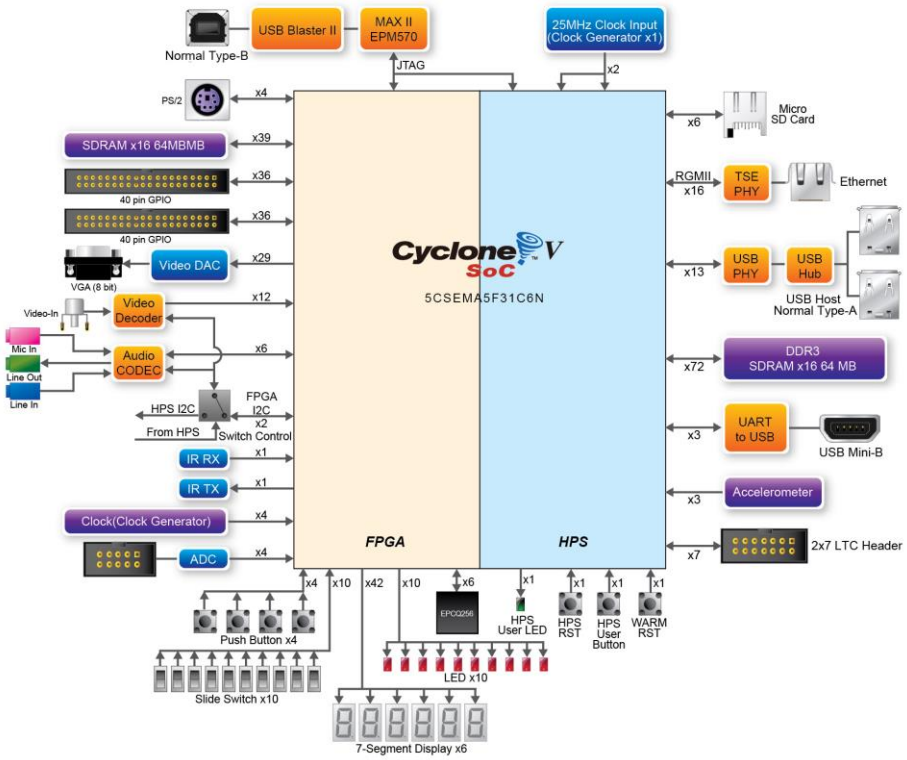


Figure 2-3 Board Block Diagram

Following is more detailed information about the blocks in **Figure 2-3**:

FPGA Device

- Cyclone V SoC 5CSEMA5F31 Device
- Dual-core ARM Cortex-A9 (HPS)
- 85K Programmable Logic Elements
- 4,450 Kbits embedded memory
- 6 Fractional PLLs
- 2 Hard Memory Controllers

Configuration and Debug

- Quad Serial Configuration device – EPCQ256 on FPGA
- On-Board USB Blaster II (Normal type B USB connector)

Memory Device

- 64MB (32Mx16) SDRAM on FPGA
- 1GB (2x256Mx16) DDR3 SDRAM on HPS
- Micro SD Card Socket on HPS

Communication

- Two Port USB 2.0 Host (ULPI interface with USB type A connector)
- UART to USB (USB Mini-B connector)
- 10/100/1000 Ethernet
- PS/2 mouse/keyboard
- IR Emitter/Receiver
- I2C Multiplexer

Connectors

- Two 40-pin Expansion Headers
- One 10-pin ADC Input Header
- One LTC connector (One Serial Peripheral Interface (SPI) Master ,one I2C and one GPIO interface)

Display

- 24-bit VGA DAC

Audio

- 24-bit CODEC, Line-in, line-out, and microphone-in jacks

Video Input

- TV Decoder (NTSC/PAL/SECAM) and TV-in connector

ADC

- Fast throughput rate: 1 MSPS
- Channel number: 8
- Resolution: 12 bits
- Analog input range : 0 ~ 2.5 V or 0 ~ 5V as selected via the RANGE bit in the control register

Switches, Buttons and Indicators

- 5 User Keys (FPGA x4, HPS x1)
- 10 User switches (FPGA x10)
- 11 User LEDs (FPGA x10, HPS x 1)
- 2 HPS Reset Buttons (HPS_RESET_n and HPS_WARM_RST_n)
- Six 7-segment displays

Sensors

- G-Sensor on HPS

Power

- 12V DC input

Chapter 3

Using the DE1-SoC Board

This chapter gives instructions for using the board and describes each of its peripherals.

3.1 FPGA Configuration Mode Setting

When the DE1-SoC board is powered on, the FPGA can be configured from EPCQ or HPS. The MSEL[4:0] pins are used to specify the configuration scheme. The MSEL[4:0] is implemented as a 6-pin DIP switch **SW10** on the DE1-SoC board, as shown in **Figure 3-1**. The dip 6 on SW10 is not used.

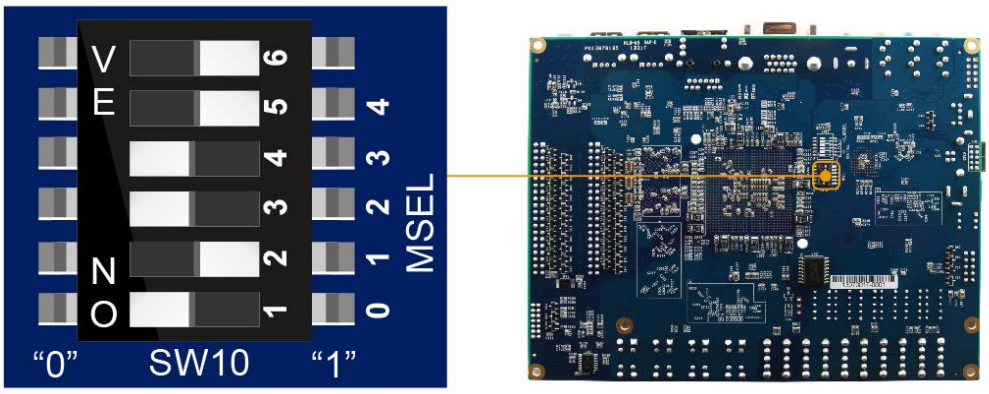


Figure 3-1 The JTAG chain on the board

Table 3-1 shows the relation between MSEL[4:0] and DIP switch in SW10.

Table 3-1 SW10 FPGA Configuration Mode Switch

Board Reference	Signal Name	Description	Default
SW10.1	MSEL0	Use these pins to set the FPGA configuration scheme.	On (“0”)
SW10.2	MSEL1		Off (“1”)
SW10.3	MSEL2		On (“0”)
SW10.4	MSEL3		On (“0”)
SW10.5	MSEL4		Off(“1”)
SW10.6	N/A	N/A	N/A

Figure 3-1 shows valid MSEL[4:0] settings for FPGA configuration mode of DE1-SoC. DE1-SoC is shipped with 'AS' mode as the default mode. When the board is powered on, the FPGA is configured from EPCQ which is pre-programmed with the manufacturer default code. If developers wish to reconfigure FPGA from an application software running on Linux, developers need to adjust MSEL[4:0] to “01010” just before the programming process. If developers using the "Linux Console with framebuffer" or "Linux LXDE Desktop" SD Card image, developers need to adjust MSEL[4:0] to “00000” before board is power on.

Table 3-2 MSEL pin Settings for FPGA Configure of DE1-SoC

MSEL[4:0]	Configure Scheme	Description
10010	AS	FPGA configured from EPCQ (default)
01010	FPPx32	FPGA configured from HPS software: Linux
00000	FPPx16	FPGA configured from HPS software: U-Boot, with image stored on the SD card, like LXDE Desktop or console Linux with framebuffer edition.

3.2 Configuring the Cyclone V SoC FPGA

The DE1-SoC board contains a serial configuration device that stores configuration data for the Cyclone V SoC FPGA. This configuration data is automatically loaded from the configuration device into the FPGA every time while power is applied to the board. Using the Quartus II software, it is possible to reconfigure the FPGA at any time, and it is also possible to change the non-volatile data that is stored in the serial configuration device. Both types of programming methods are described below:

1. JTAG programming: In this method of programming, named after the IEEE standards Joint Test Action Group, the configuration bit stream is downloaded directly into the Cyclone V SoC FPGA. The FPGA will retain this configuration as long as power is applied to the board; the configuration information will be lost when the power is turned off.

2. AS programming: In this method, called Active Serial programming, the configuration bit stream is downloaded into the quad serial configuration device (EPCQ256). It provides non-volatile storage of the bit stream, so that the information is retained even when the power supply to the DE1-SoC board is turned off. When the board's power is turned on, the configuration data in the EPCQ256 device is automatically loaded into the Cyclone V SoC FPGA.

■ JTAG Chain on DE1-SoC Board

To use JTAG interface for configuring the FPGA device, the JTAG chain on DE1-SoC must form a closed loop that allows Quartus II programmer to detect FPGA device. **Figure 3-2** illustrates the JTAG chain on DE1-SoC board.

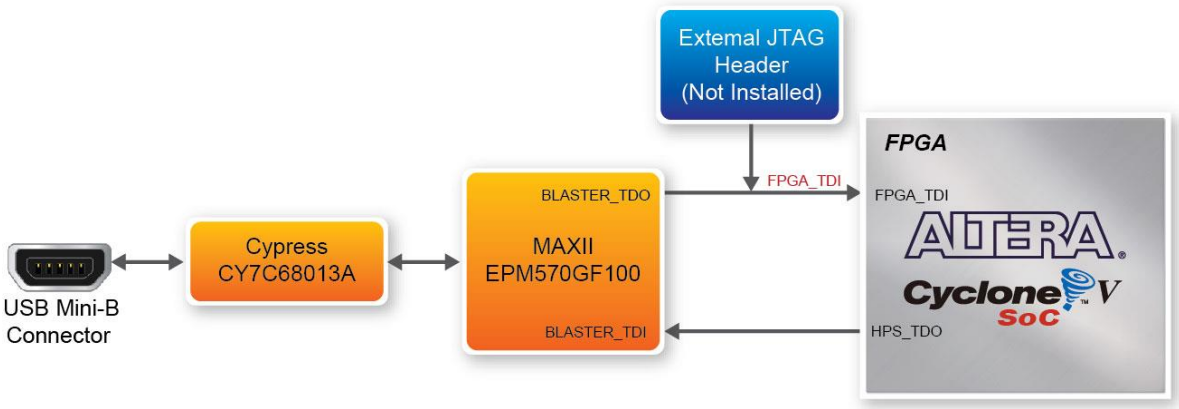


Figure 3-2 The JTAG chain on the board

■ Configuring the FPGA in JTAG Mode

There are two devices (FPGA and HPS) on the JTAG Chain, the configure flow is different from the one used with DE1. The following shows the programming flow with JTAG mode step by step.

- Open Programmer and click "Auto Detect " as **Figure 3-3**

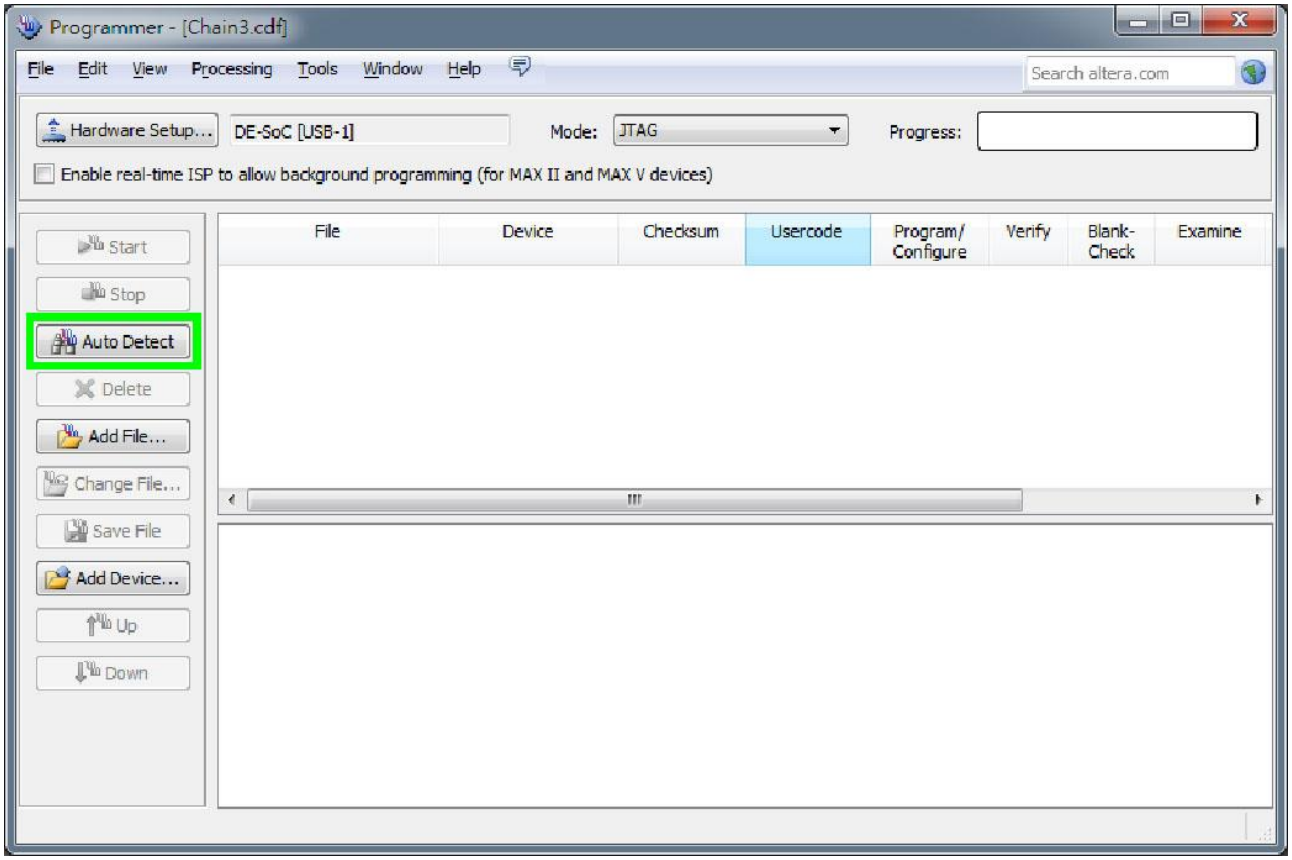


Figure 3-3 FPGA JTAG Programming Steps 1

Select detected device as **Figure 3-4** (Please select the device associated with the board)

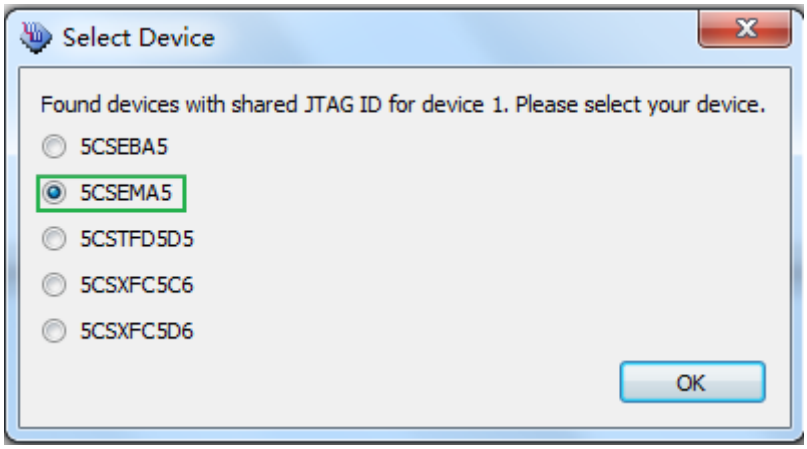


Figure 3-4 FPGA JTAG Programming Steps 2

- Both FPGA and HPS will be detected as **Figure 3-5**

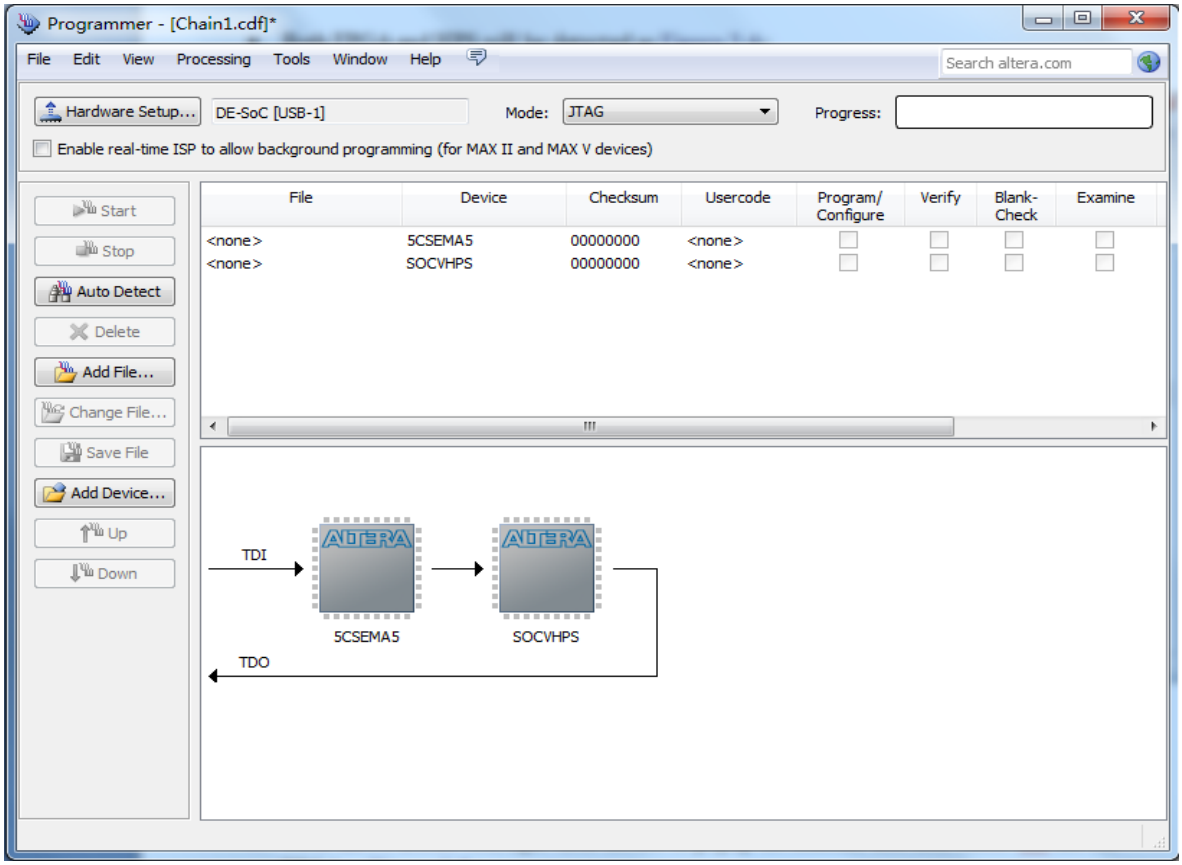


Figure 3-5 FPGA JTAG Programming Steps 3

- Click the FPGA device, right click mouse to popup the manual, and then select .sof file for FPGA as **Figure 3-6**

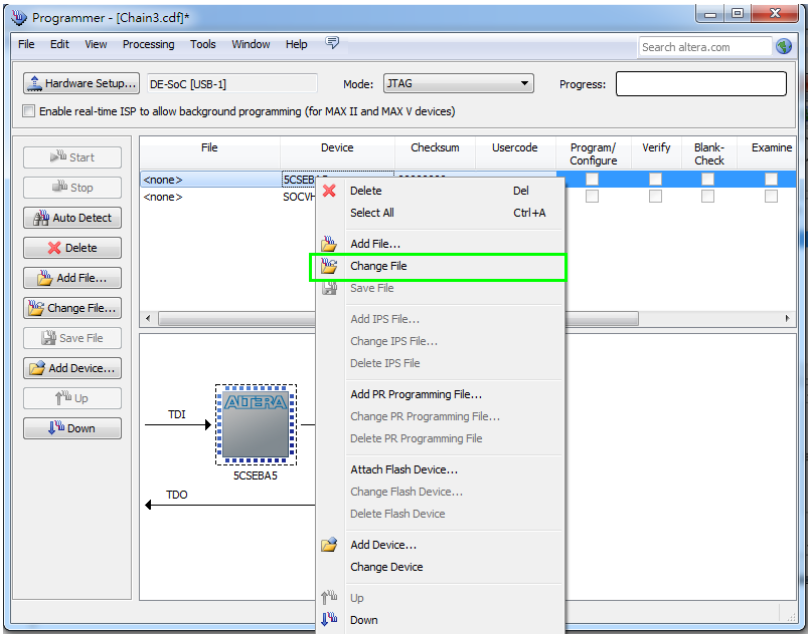


Figure 3-6 FPGA JTAG Programming Steps 4

- Select .sof file for FPGA as **Figure 3-7**

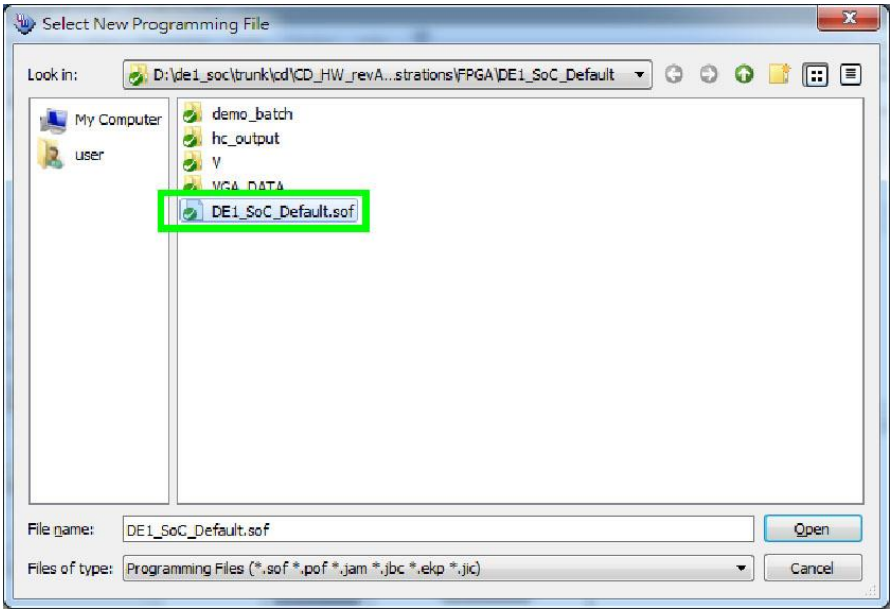


Figure 3-7 FPGA JTAG Programming Steps 5

- Click “Program/Configure” check box, and then click “Start” button to download .sof file into FPGA as **Figure 3-8**

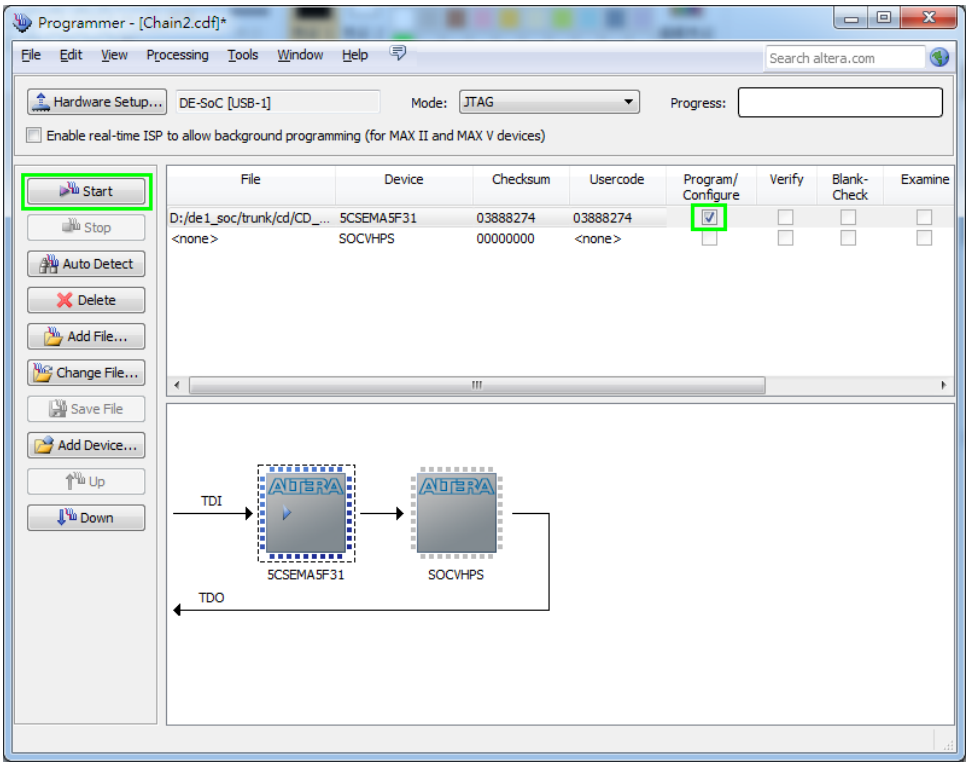


Figure 3-8 FPGA JTAG Programming Steps 6

■ Configuring the FPGA in AS Mode (from EPCQ256)

- The board contains a quad serial configuration device (EPCQ256) that stores configuration data for the Cyclone V SoC FPGA. This configuration data is automatically loaded from the quad serial configuration device chip into the FPGA when the board is powered up.
- To program the configuration device, users will need to use a Serial Flash Loader (SFL) function to program the quad serial configuration device via the JTAG interface. The FPGA-based SFL is a soft intellectual property (IP) core within the FPGA that bridges the JTAG and flash interfaces. The SFL mega-function is available from Quartus II software. **Figure 3-9** shows the programming method when adopting a SFL solution
- Please refer to Chapter 9: Steps of Programming the Quad Serial Configuration Device for the basic programming instruction on the serial configuration device

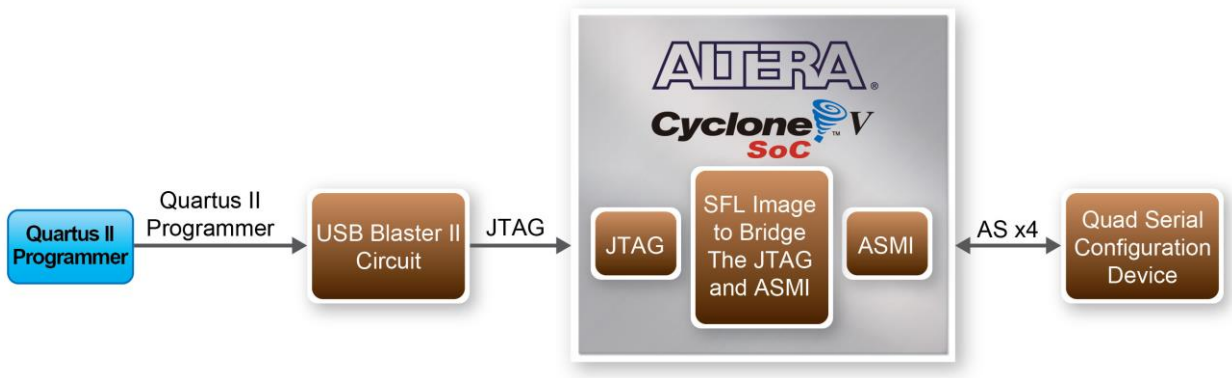


Figure 3-9 Programming a Quad Serial Configuration Device with the SFL Solution

3.3 Board Status Elements

The board includes status LEDs. Please refer t**o** Table 3-3 for the status of the LED indicator.

Table 3-3 LED Indicators

Board Reference	*LED Name*	*Description*
D14	12-V Power	Illuminates when 12-V power is active.
TXD	UART TXD	Illuminates when data from FT232R to USB Host.
RXD	UART RXD	Illuminates when data from USB Host to FT232R.
D5	JTAG_RX	Reserved
D4	JTAG_TX	

3.4 Board Reset Elements

The board equips two HPS reset circuits (See Figure 3-10). Table 3-4 shows the buttons references and its descriptions. Figure 3-11 shows the reset tree on the board.

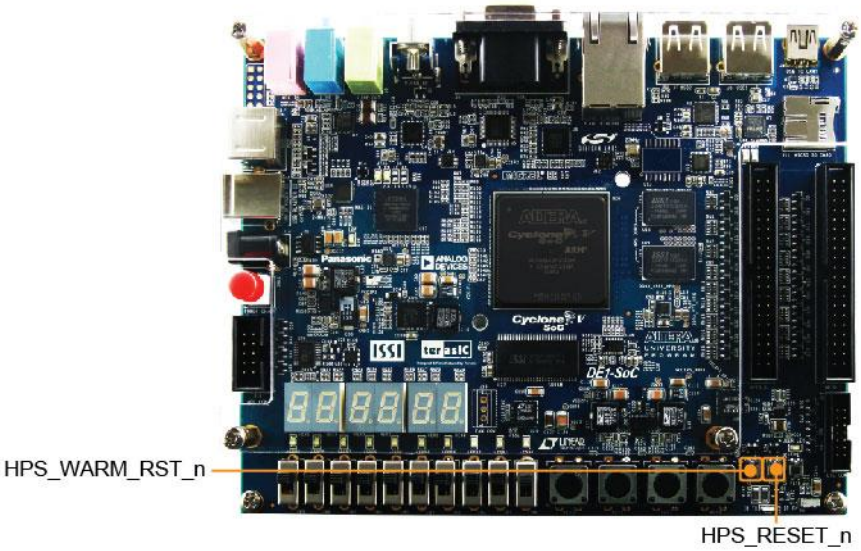


Figure 3-10 Board Reset Elements

Table 3-4 Reset Elements

Board Reference	*Signal Name*	*Description*
KEY5	HPS_RESET_N	Cold reset to the HPS, Ethernet PHY and USB host device . Active low input that will reset all HPS logics that can be reset.
KEY7	HPS_WARM_RST_N	Warm reset to the HPS block. Active low input affects the system reset domains which allows debugging to operate.

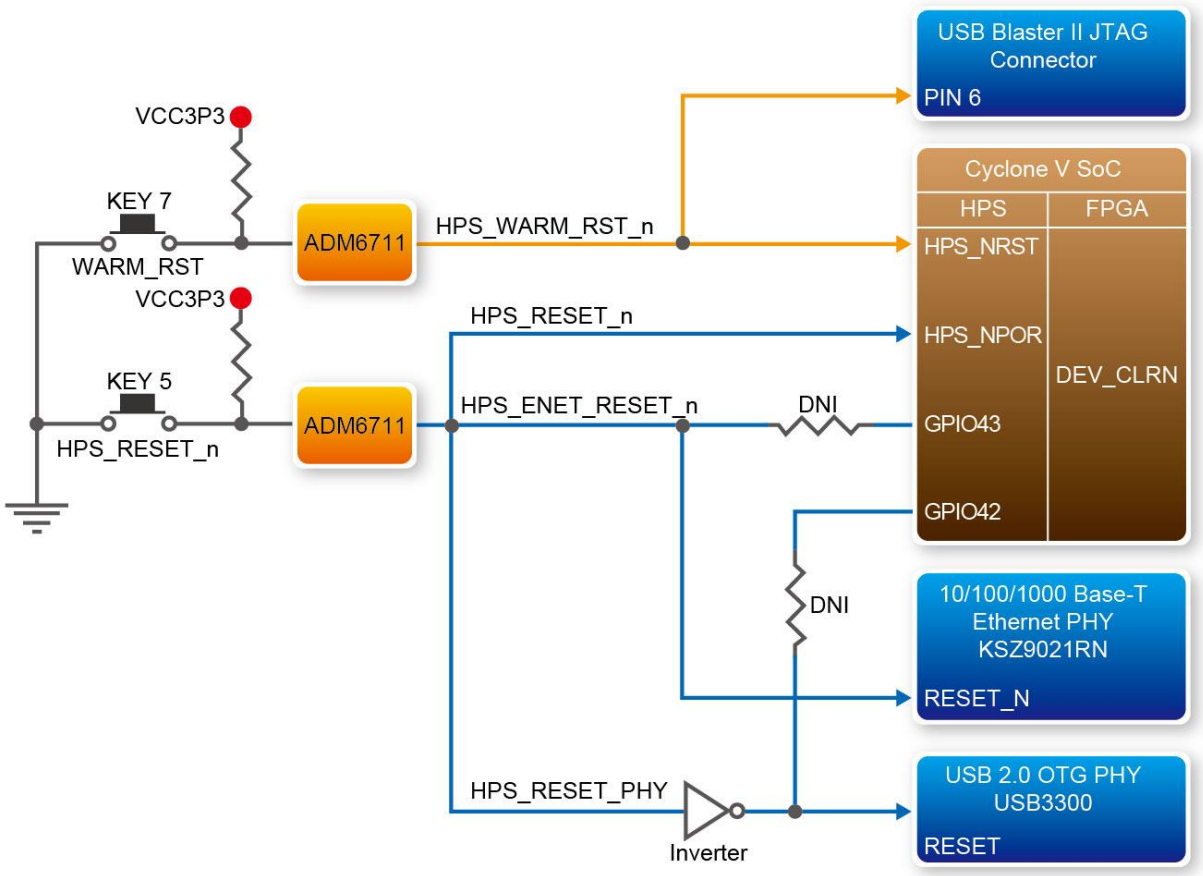


Figure 3-11 Reset Tree on the Development Board

3.5 Clock Circuitry

Figure 3-12 is a diagram showing the default frequencies of all of the external clocks going to the Cyclone V SoC FPGA. A clock generator is used to distribute clock signals with low jitter to FPGA. The four distributing 50MHz clock signals are connected to the FPGA that are used for clocking the user logic. One distributing 25MHz clock signal is connected to HPS clock inputs, the other distributing 25MHz clock signal is connected to the clock input of Gigabit Ethernet Transceiver. Two distributing 24MHz clock signals are connected to clock inputs of USB Host/OTG PHY and USB Hub controller, respectively. The associated pin assignments for clock inputs to FPGA I/O pins are listed in Table 3-5.

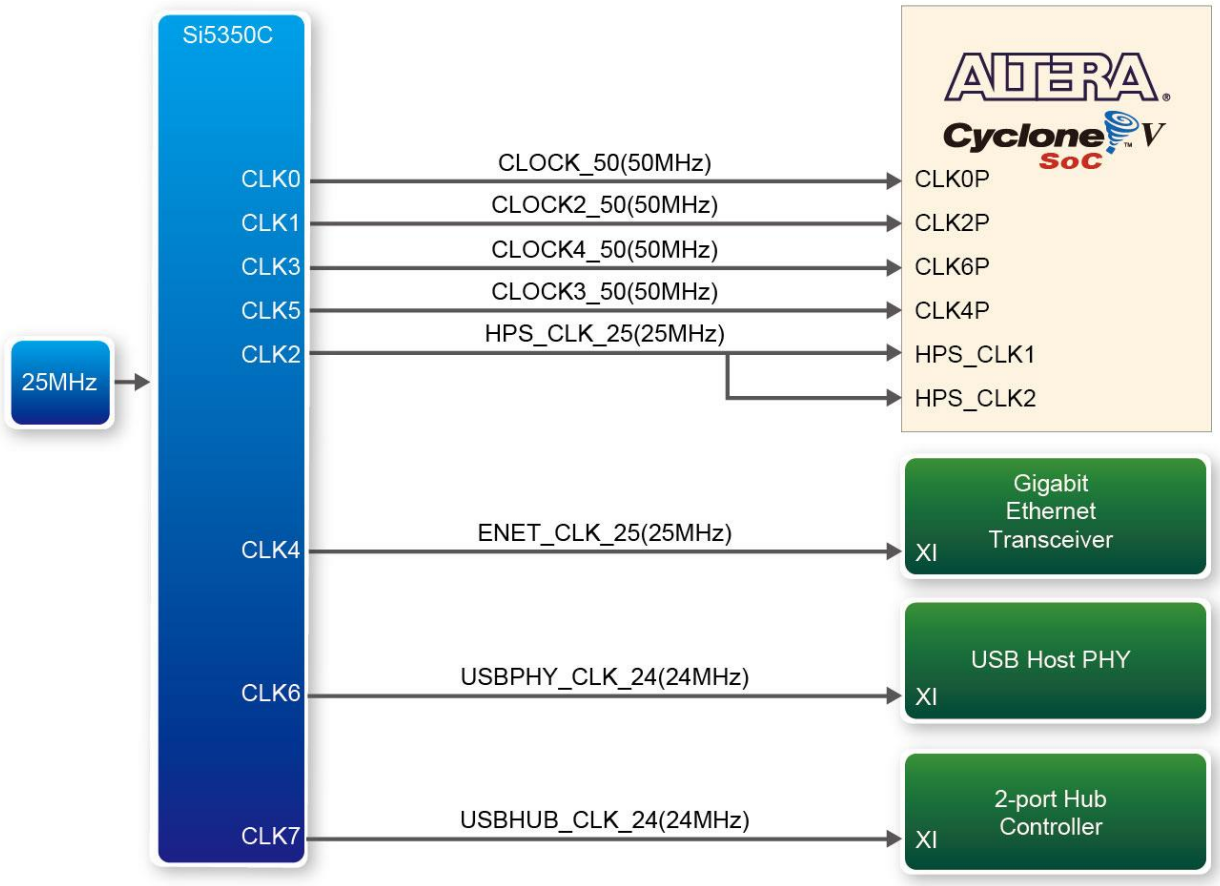


Figure 3-12 Block diagram of the clock distribution

Table 3-5 Pin Assignments for Clock Inputs

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
CLOCK_50	**PIN_AF14**	**50 MHz clock input**	**3.3V**
CLOCK2_50	**PIN_AA16**	**50 MHz clock input**	**3.3V**
CLOCK3_50	**PIN_Y26**	**50 MHz clock input**	**3.3V**
CLOCK4_50	**PIN_K14**	**50 MHz clock input**	**3.3V**
HPS_CLOCK1_25	**PIN_D25**	**25 MHz clock input**	**3.3V**
HPS_CLOCK2_25	**PIN_F25**	**25 MHz clock input**	**3.3V**

3.6 Interface on FPGA

This section describes the interfaces to the FPGA. Users can control or monitor the different interfaces with user logic on the FPGA.

3.6.1 User Push-buttons, Switches and LEDs on FPGA

The board provides four push-button switches connected to FPGA as shown in **Figure 3-13** Connections between the push-button and Cyclone V SoC FPGA. Each of these switches is debounced using a Schmitt Trigger circuit, as indicated in **Figure 3-14**. The four outputs called KEY0, KEY1, KEY2, and KEY3 of the Schmitt Trigger devices are connected directly to the Cyclone V SoC FPGA. Each push-button switch provides a high logic level when it is not pressed, and provides a low logic level when depressed. Since the push-button switches are debounced, they are appropriate for using as clock or reset inputs in a circuit.

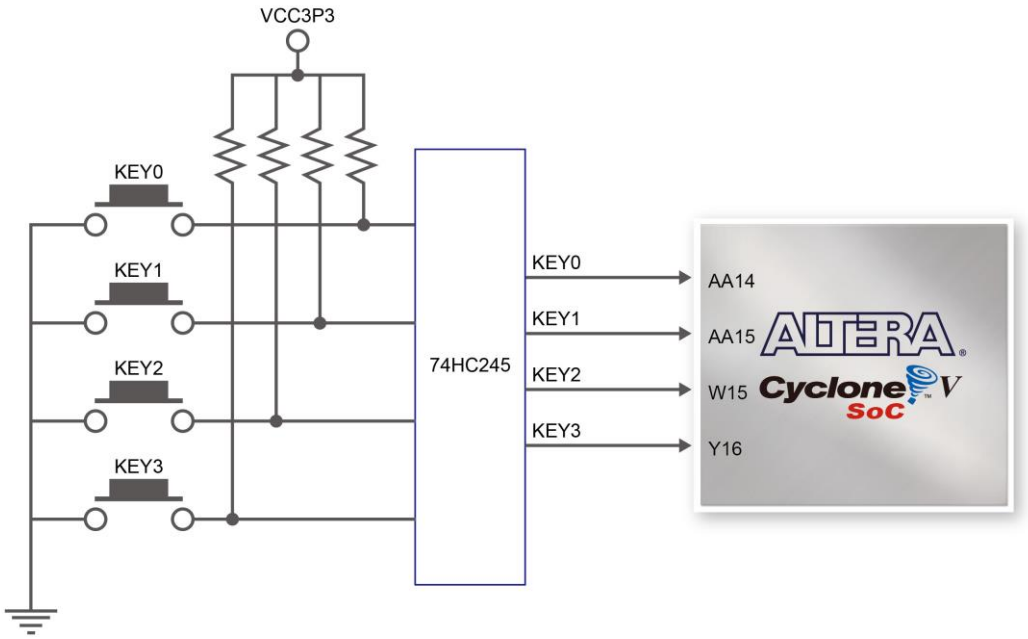


Figure 3-13 Connections between the push-button and Cyclone V SoC FPGA

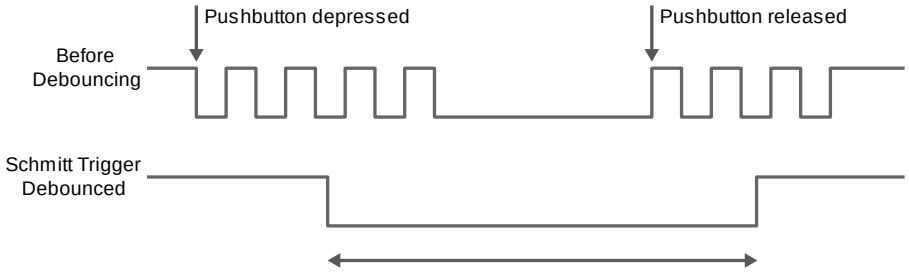


Figure 3-14 Switch debouncing

There are ten slide switches connected to FPGA on the board (See **Figure 3-15**). These switches are not debounced, and are assumed for use as level-sensitive data inputs to a circuit. Each switch is connected directly to a pin on the Cyclone V SoC FPGA. When the switch is in the DOWN position (closest to the edge of the board), it provides a low logic level to the FPGA, and when the switch is in the UP position it provides a high logic level.

Figure 3-15 Connections between the slide switches and Cyclone V SoC FPGA

There are also ten user-controllable LEDs connected to FPGA on the board. Each LED is driven directly by a pin on the Cyclone V SoC FPGA; driving its associated pin to a high logic level turns the LED on, and driving the pin low turns it off. **Figure 3-16** shows the connections between LEDs and Cyclone V SoC FPGA. **Table 3-6**, **Table 3-7** and **Table 3-8** list the pin assignments of these user interfaces.

Figure 3-16 Connections between the LEDs and Cyclone V SoC FPGA

Table 3-6 Pin Assignments for Slide Switches

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
SW[0]	PIN_AB12	Slide Switch[0]	3.3V
SW[1]	PIN_AC12	Slide Switch[1]	3.3V
SW[2]	PIN_AF9	Slide Switch[2]	3.3V
SW[3]	PIN_AF10	Slide Switch[3]	3.3V
SW[4]	PIN_AD11	Slide Switch[4]	3.3V
SW[5]	PIN_AD12	Slide Switch[5]	3.3V
SW[6]	PIN_AE11	Slide Switch[6]	3.3V
SW[7]	PIN_AC9	Slide Switch[7]	3.3V
SW[8]	PIN_AD10	Slide Switch[8]	3.3V
SW[9]	PIN_AE12	Slide Switch[9]	3.3V

Table 3-7 Pin Assignments for Push-buttons

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
KEY[0]	PIN_AA14	Push-button[0]	3.3V
KEY[1]	PIN_AA15	Push-button[1]	3.3V
KEY[2]	PIN_W15	Push-button[2]	3.3V
KEY[3]	PIN_Y16	Push-button[3]	3.3V

Table 3-8 Pin Assignments for LEDs

Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR[0]	PIN_V16	LED [0]	3.3V
LEDR[1]	PIN_W16	LED [1]	3.3V
LEDR[2]	PIN_V17	LED [2]	3.3V
LEDR[3]	PIN_V18	LED [3]	3.3V
LEDR[4]	PIN_W17	LED [4]	3.3V
LEDR[5]	PIN_W19	LED [5]	3.3V
LEDR[6]	PIN_Y19	LED [6]	3.3V
LEDR[7]	PIN_W20	LED [7]	3.3V
LEDR[8]	PIN_W21	LED [8]	3.3V
LEDR[9]	PIN_Y21	LED [9]	3.3V

3.6.2 Using the 7-segment Displays

The DE1-SoC board has six 7-segment displays. These displays are arranged into three pairs, meant for displaying numbers of various sizes. As indicated in the schematic in Figure 3-17, the seven segments (common anode) are connected to pins on Cyclone V SoC FPGA. Applying a low logic level to a segment will light it up and applying a high logic level turns it off.

Each segment in a display is identified by an index from 0 to 6, with the positions given in Figure 3-17.Table 3-9 shows the assignments of FPGA pins to the 7-segment displays.

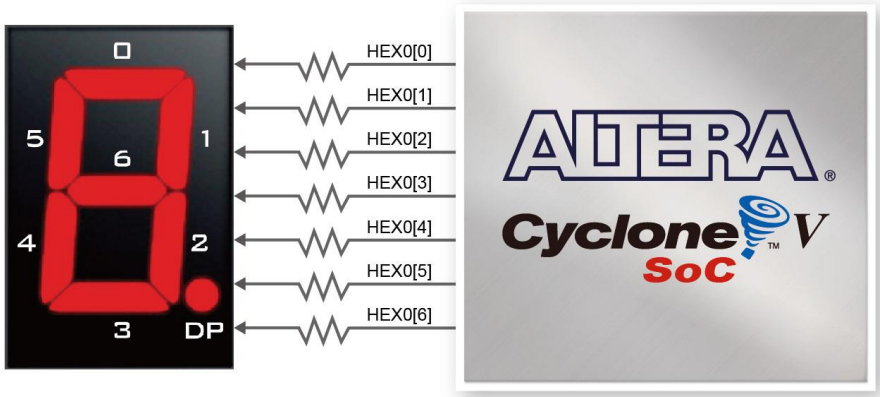


Figure 3-17 Connections between the 7-segment display HEX0 and Cyclone V SoC FPGA

Table 3-9 Pin Assignments for 7-segment Displays

Signal Name	FPGA Pin No.	Description	I/O Standard
HEX0[0]	PIN_AE26	Seven Segment Digit 0[0]	3.3V

HEX0[1]	PIN_AE27	Seven Segment Digit 0[1]	3.3V
HEX0[2]	PIN_AE28	Seven Segment Digit 0[2]	3.3V
HEX0[3]	PIN_AG27	Seven Segment Digit 0[3]	3.3V
HEX0[4]	PIN_AF28	Seven Segment Digit 0[4]	3.3V
HEX0[5]	PIN_AG28	Seven Segment Digit 0[5]	3.3V
HEX0[6]	PIN_AH28	Seven Segment Digit 0[6]	3.3V
HEX1[0]	PIN_AJ29	Seven Segment Digit 1[0]	3.3V
HEX1[1]	PIN_AH29	Seven Segment Digit 1[1]	3.3V
HEX1[2]	PIN_AH30	Seven Segment Digit 1[2]	3.3V
HEX1[3]	PIN_AG30	Seven Segment Digit 1[3]	3.3V
HEX1[4]	PIN_AF29	Seven Segment Digit 1[4]	3.3V
HEX1[5]	PIN_AF30	Seven Segment Digit 1[5]	3.3V
HEX1[6]	PIN_AD27	Seven Segment Digit 1[6]	3.3V
HEX2[0]	PIN_AB23	Seven Segment Digit 2[0]	3.3V
HEX2[1]	PIN_AE29	Seven Segment Digit 2[1]	3.3V
HEX2[2]	PIN_AD29	Seven Segment Digit 2[2]	3.3V
HEX2[3]	PIN_AC28	Seven Segment Digit 2[3]	3.3V
HEX2[4]	PIN_AD30	Seven Segment Digit 2[4]	3.3V
HEX2[5]	PIN_AC29	Seven Segment Digit 2[5]	3.3V
HEX2[6]	PIN_AC30	Seven Segment Digit 2[6]	3.3V
HEX3[0]	PIN_AD26	Seven Segment Digit 3[0]	3.3V
HEX3[1]	PIN_AC27	Seven Segment Digit 3[1]	3.3V
HEX3[2]	PIN_AD25	Seven Segment Digit 3[2]	3.3V
HEX3[3]	PIN_AC25	Seven Segment Digit 3[3]	3.3V
HEX3[4]	PIN_AB28	Seven Segment Digit 3[4]	3.3V
HEX3[5]	PIN_AB25	Seven Segment Digit 3[5]	3.3V
HEX3[6]	PIN_AB22	Seven Segment Digit 3[6]	3.3V
HEX4[0]	PIN_AA24	Seven Segment Digit 4[0]	3.3V
HEX4[1]	PIN_Y23	Seven Segment Digit 4[1]	3.3V
HEX4[2]	PIN_Y24	Seven Segment Digit 4[2]	3.3V
HEX4[3]	PIN_W22	Seven Segment Digit 4[3]	3.3V
HEX4[4]	PIN_W24	Seven Segment Digit 4[4]	3.3V
HEX4[5]	PIN_V23	Seven Segment Digit 4[5]	3.3V
HEX4[6]	PIN_W25	Seven Segment Digit 4[6]	3.3V
HEX5[0]	PIN_V25	Seven Segment Digit 5[0]	3.3V
HEX5[1]	PIN_AA28	Seven Segment Digit 5[1]	3.3V
HEX5[2]	PIN_Y27	Seven Segment Digit 5[2]	3.3V
HEX5[3]	PIN_AB27	Seven Segment Digit 5[3]	3.3V
HEX5[4]	PIN_AB26	Seven Segment Digit 5[4]	3.3V
HEX5[5]	PIN_AA26	Seven Segment Digit 5[5]	3.3V
HEX5[6]	PIN_AA25	Seven Segment Digit 5[6]	3.3V

3.6.3 Using the 2x20 GPIO Expansion Headers

The board provides two 40-pin expansion headers. The header connects directly to 36 pins of the Cyclone V SoC FPGA, and also provides DC +5V (VCC5), DC +3.3V (VCC3P3), and two GND pins. The maximum power consumption of the daughter card that connects to GPIO port is shown in Table 3-10.

Table 3-10 Power Supply of the Expansion Header

Supplied Voltage	*Max. Current Limit*
5V	1A
3.3V	1.5A

Each pin on the expansion headers is connected to two diodes and a resistor that provides protection against high and low voltages. Figure 3-18 shows the protection circuitry for only one of the pin on the header, but this circuitry is included for all 72 data pins. Table 3-11 shows all the pin assignments of the GPIO connector.

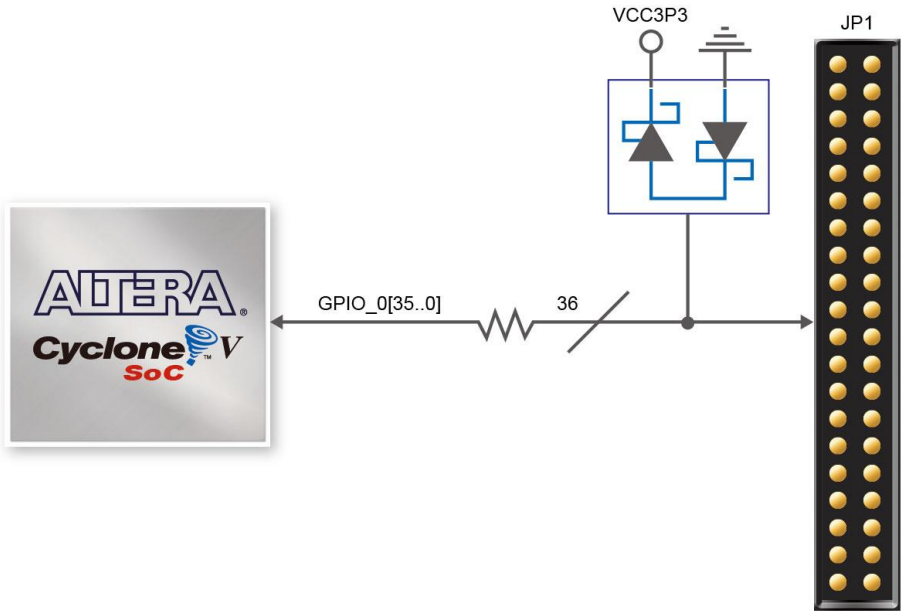


Figure 3-18 Connections between the GPIO connector and Cyclone V SoC FPGA

Table 3-11 Pin Assignments for Expansion Headers

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
GPIO_0[0]	PIN_AC18	GPIO Connection 0[0]	3.3V
GPIO_0 [1]	PIN_Y17	GPIO Connection 0[1]	3.3V
GPIO_0 [2]	PIN_AD17	GPIO Connection 0[2]	3.3V
GPIO_0 [3]	PIN_Y18	GPIO Connection 0[3]	3.3V

GPIO_0 [4]	PIN_AK16	GPIO Connection 0[4]	3.3V
GPIO_0 [5]	PIN_AK18	GPIO Connection 0[5]	3.3V
GPIO_0 [6]	PIN_AK19	GPIO Connection 0[6]	3.3V
GPIO_0 [7]	PIN_AJ19	GPIO Connection 0[7]	3.3V
GPIO_0 [8]	PIN_AJ17	GPIO Connection 0[8]	3.3V
GPIO_0 [9]	PIN_AJ16	GPIO Connection 0[9]	3.3V
GPIO_0 [10]	PIN_AH18	GPIO Connection 0[10]	3.3V
GPIO_0 [11]	PIN_AH17	GPIO Connection 0[11]	3.3V
GPIO_0 [12]	PIN_AG16	GPIO Connection 0[12]	3.3V
GPIO_0 [13]	PIN_AE16	GPIO Connection 0[13]	3.3V
GPIO_0 [14]	PIN_AF16	GPIO Connection 0[14]	3.3V
GPIO_0 [15]	PIN_AG17	GPIO Connection 0[15]	3.3V
GPIO_0 [16]	PIN_AA18	GPIO Connection 0[16]	3.3V
GPIO_0 [17]	PIN_AA19	GPIO Connection 0[17]	3.3V
GPIO_0 [18]	PIN_AE17	GPIO Connection 0[18]	3.3V
GPIO_0 [19]	PIN_AC20	GPIO Connection 0[19]	3.3V
GPIO_0 [20]	PIN_AH19	GPIO Connection 0[20]	3.3V
GPIO_0 [21]	PIN_AJ20	GPIO Connection 0[21]	3.3V
GPIO_0 [22]	PIN_AH20	GPIO Connection 0[22]	3.3V
GPIO_0 [23]	PIN_AK21	GPIO Connection 0[23]	3.3V
GPIO_0 [24]	PIN_AD19	GPIO Connection 0[24]	3.3V
GPIO_0 [25]	PIN_AD20	GPIO Connection 0[25]	3.3V
GPIO_0 [26]	PIN_AE18	GPIO Connection 0[26]	3.3V
GPIO_0 [27]	PIN_AE19	GPIO Connection 0[27]	3.3V
GPIO_0 [28]	PIN_AF20	GPIO Connection 0[28]	3.3V
GPIO_0 [29]	PIN_AF21	GPIO Connection 0[29]	3.3V
GPIO_0 [30]	PIN_AF19	GPIO Connection 0[30]	3.3V
GPIO_0 [31]	PIN_AG21	GPIO Connection 0[31]	3.3V
GPIO_0 [32]	PIN_AF18	GPIO Connection 0[32]	3.3V
GPIO_0 [33]	PIN_AG20	GPIO Connection 0[33]	3.3V
GPIO_0 [34]	PIN_AG18	GPIO Connection 0[34]	3.3V
GPIO_0 [35]	PIN_AJ21	GPIO Connection 0[35]	3.3V
GPIO_1[0]	PIN_AB17	GPIO Connection 1[0]	3.3V
GPIO_1[1]	PIN_AA21	GPIO Connection 1[1]	3.3V
GPIO_1 [2]	PIN_AB21	GPIO Connection 1[2]	3.3V
GPIO_1 [3]	PIN_AC23	GPIO Connection 1[3]	3.3V
GPIO_1 [4]	PIN_AD24	GPIO Connection 1[4]	3.3V
GPIO_1 [5]	PIN_AE23	GPIO Connection 1[5]	3.3V
GPIO_1 [6]	PIN_AE24	GPIO Connection 1[6]	3.3V
GPIO_1 [7]	PIN_AF25	GPIO Connection 1[7]	3.3V
GPIO_1 [8]	PIN_AF26	GPIO Connection 1[8]	3.3V
GPIO_1 [9]	PIN_AG25	GPIO Connection 1[9]	3.3V
GPIO_1[10]	PIN_AG26	GPIO Connection 1[10]	3.3V
GPIO_1 [11]	PIN_AH24	GPIO Connection 1[11]	3.3V

GPIO_1 [12]	PIN_AH27	GPIO Connection 1[12]	3.3V
GPIO_1 [13]	PIN_AJ27	GPIO Connection 1[13]	3.3V
GPIO_1 [14]	PIN_AK29	GPIO Connection 1[14]	3.3V
GPIO_1 [15]	PIN_AK28	GPIO Connection 1[15]	3.3V
GPIO_1 [16]	PIN_AK27	GPIO Connection 1[16]	3.3V
GPIO_1 [17]	PIN_AJ26	GPIO Connection 1[17]	3.3V
GPIO_1 [18]	PIN_AK26	GPIO Connection 1[18]	3.3V
GPIO_1 [19]	PIN_AH25	GPIO Connection 1[19]	3.3V
GPIO_1 [20]	PIN_AJ25	GPIO Connection 1[20]	3.3V
GPIO_1 [21]	PIN_AJ24	GPIO Connection 1[21]	3.3V
GPIO_1 [22]	PIN_AK24	GPIO Connection 1[22]	3.3V
GPIO_1 [23]	PIN_AG23	GPIO Connection 1[23]	3.3V
GPIO_1 [24]	PIN_AK23	GPIO Connection 1[24]	3.3V
GPIO_1 [25]	PIN_AH23	GPIO Connection 1[25]	3.3V
GPIO_1 [26]	PIN_AK22	GPIO Connection 1[26]	3.3V
GPIO_1 [27]	PIN_AJ22	GPIO Connection 1[27]	3.3V
GPIO_1 [28]	PIN_AH22	GPIO Connection 1[28]	3.3V
GPIO_1 [29]	PIN_AG22	GPIO Connection 1[29]	3.3V
GPIO_1 [30]	PIN_AF24	GPIO Connection 1[30]	3.3V
GPIO_1 [31]	PIN_AF23	GPIO Connection 1[31]	3.3V
GPIO_1 [32]	PIN_AE22	GPIO Connection 1[32]	3.3V
GPIO_1 [33]	PIN_AD21	GPIO Connection 1[33]	3.3V
GPIO_1 [34]	PIN_AA20	GPIO Connection 1[34]	3.3V
GPIO_1 [35]	PIN_AC22	GPIO Connection 1[35]	3.3V

3.6.4 Using the 24-bit Audio CODEC

The DE1-SoC board provides high-quality 24-bit audio via the Wolfson WM8731 audio CODEC (Encoder/Decoder). This chip supports microphone-in, line-in, and line-out ports, with a sample rate adjustable from 8 kHz to 96 kHz. The WM8731 is controlled via a serial I2C bus, which is connected to HPS or Cyclone V SoC FPGA through a I2C multiplexer. A schematic diagram of the audio circuitry is shown in **Figure 3-19**, and the FPGA pin assignments are listed in **Table 3-12**. Detailed information for using the WM8731 codec is available in its datasheet, which can be found on the manufacturer's website, or in the DE1_SOC_datasheets\Audio CODEC folder on the DE1-SoC System CD.

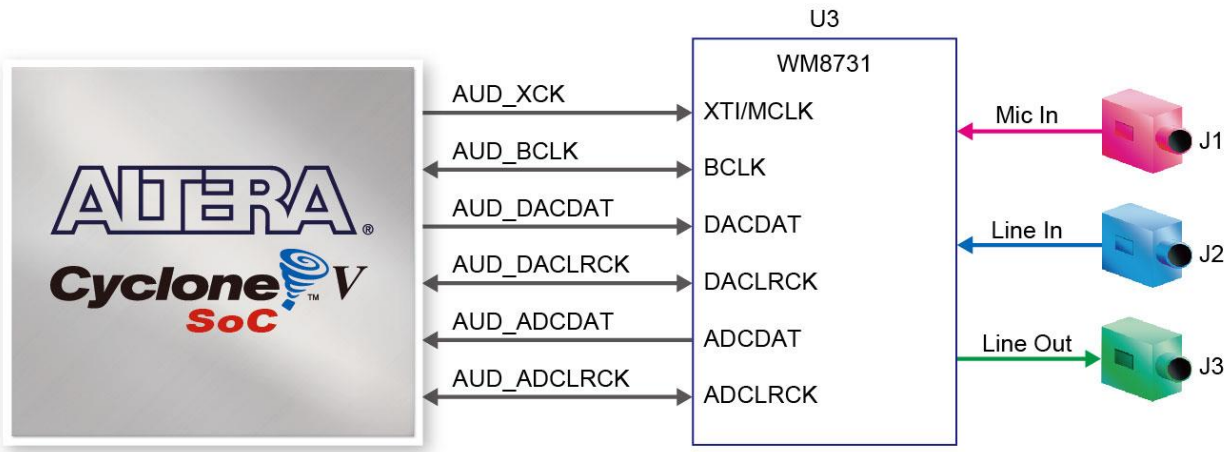


Figure 3-19 Connections between FPGA and Audio CODEC

Table 3-12 Audio CODEC Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
AUD_ADCLRCK	PIN_K8	Audio CODEC ADC LR Clock	3.3V
AUD_ADCDAT	PIN_K7	Audio CODEC ADC Data	3.3V
AUD_DACLRCK	PIN_H8	Audio CODEC DAC LR Clock	3.3V
AUD_DACDAT	PIN_J7	Audio CODEC DAC Data	3.3V
AUD_XCK	PIN_G7	Audio CODEC Chip Clock	3.3V
AUD_BCLK	PIN_H7	Audio CODEC Bit-Stream Clock	3.3V
I2C_SCLK	PIN_J12 or PIN_E23	I2C Clock	3.3V
I2C_SDAT	PIN_K12 or PIN_C24	I2C Data	3.3V

3.6.5 I2C Multiplexer

The DE1-SoC board implements an I2C multiplexer so that HPS can access the I2C bus originally owned by FPGA. **Figure 3-20** shows the connection of I2C multiplexer. HPS will own I2C bus and then can access Audio CODEC and TV Decoder when the HPS_I2C_CONTROL signal is set to high. By default, FPGA owns the I2C bus. The FPGA pin assignments of I2C bus are listed in **Table 3-13** .

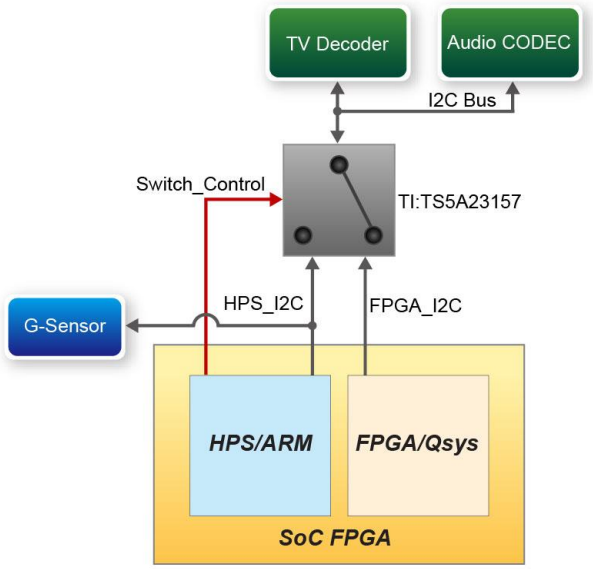


Figure 3-20 Connections of I2C Multiplexer

Table 3-13 I2C Bus Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
FPGA_I2C_SCLK	PIN_J12	FPGA I2C Clock	3.3V
FPGA_I2C_SDAT	PIN_K12	FPGA I2C Data	3.3V
HPS_I2C1_SCLK	PIN_E23	I2C Clock of the first HPS I2C concontroller	3.3V
HPS_I2C1_SDAT	PIN_C24	I2C Data of the first HPS I2C concontroller	3.3V
HPS_I2C2_SCLK	PIN_H23	I2C Clock of the second HPS I2C concontroller	3.3V
HPS_I2C2_SDAT	PIN_A25	I2C Data of the second HPS I2C concontroller	3.3V

3.6.6 VGA

The DE1-SoC board includes a 15-pin D-SUB connector for VGA output. The VGA synchronization signals are provided directly from the Cyclone V SoC FPGA, and the Analog Devices ADV7123 triple 10-bit high-speed video DAC (only the higher 8-bits are used) is used to produce the analog data signals (red, green, and blue). It could support the SXGA standard (1280*1024) with a bandwidth of 100MHz. Figure 3-21 gives the associated schematic.

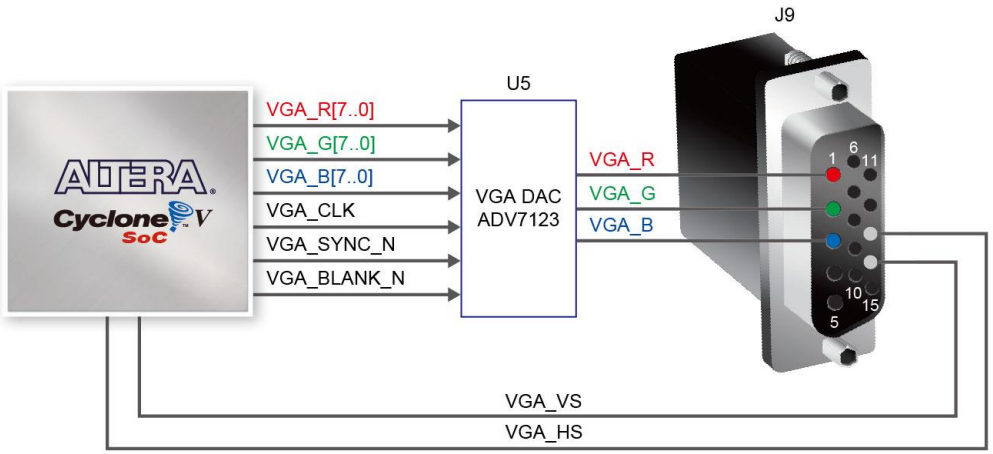


Figure 3-21 VGA Connections between FPGA and VGA

The timing specification for VGA synchronization and RGB (red, green, blue) data can be found on various educational website (for example, search for "VGA signal timing"). **Figure 3-21** illustrates the basic timing requirements for each row (horizontal) that is displayed on a VGA monitor. An active-low pulse of specific duration (time (a) in the figure) is applied to the horizontal synchronization (hsync) input of the monitor, which signifies the end of one row of data and the start of the next. The data (RGB) output to the monitor must be off (driven to 0 V) for a time period called the back porch (b) after the hsync pulse occurs, which is followed by the display interval (c). During the data display interval the RGB data drives each pixel in turn across the row being displayed. Finally, there is a time period called the front porch (d) where the RGB signals must again be off before the next hsync pulse can occur. The timing of the vertical synchronization (vsync) is the similar as shown in **Figure 3-22**, except that a vsync pulse signifies the end of one frame and the start of the next, and the data refers to the set of rows in the frame (horizontal timing). **Table 3-14** and **Table 3-15** show different resolutions and durations of time periods a, b, c, and d for both horizontal and vertical timing.

Detailed information for using the ADV7123 video DAC is available in its datasheet, which can be found on the manufacturer's website, or in the Datasheets\VIDEO DAC folder on the DE1-SoC System CD. The pin assignments between the Cyclone V SoC FPGA and the ADV7123 are listed in **Table 3-16.**

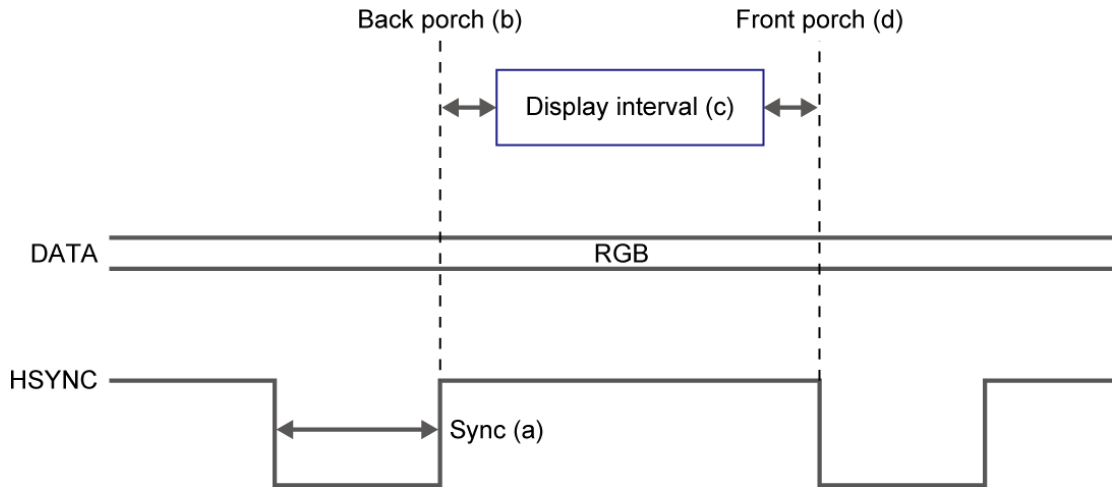


Figure 3-22 VGA horizontal timing specification

Table 3-14 VGA Horizontal Timing Specification

VGA mode		*Horizontal Timing Spec*				
Configuration	*Resolution(HxV)*	*a(us)*	*b(us)*	*c(us)*	*d(us)*	*Pixel clock(MHz)*
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36
SVGA(60Hz)	800x600	3.2	2.2	20	1	40
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56
XGA(60Hz)	1024x768	2.1	2.5	15.8	0.4	65
XGA(70Hz)	1024x768	1.8	1.9	13.7	0.3	75
XGA(85Hz)	1024x768	1.0	2.2	10.8	0.5	95
1280x1024(60Hz)	1280x1024	1.0	2.3	11.9	0.4	108

Table 3-15 VGA Vertical Timing Specification

VGA mode		*Vertical Timing Spec*				
Configuration	*Resolution(HxV)*	*a(lines)*	*b(lines)*	*c(lines)*	*d(lines)*	*Pixel clock(MHz)*
VGA(60Hz)	640x480	2	33	480	10	25
VGA(85Hz)	640x480	3	25	480	1	36
SVGA(60Hz)	800x600	4	23	600	1	40
SVGA(75Hz)	800x600	3	21	600	1	49
SVGA(85Hz)	800x600	3	27	600	1	56
XGA(60Hz)	1024x768	6	29	768	3	65
XGA(70Hz)	1024x768	6	29	768	3	75
XGA(85Hz)	1024x768	3	36	768	1	95
1280x1024(60Hz)	1280x1024	3	38	1024	1	108

Table 3-16 Pin Assignments for VGA

Signal Name	FPGA Pin No.	Description	I/O Standard
VGA_R[0]	PIN_A13	VGA Red[0]	3.3V
VGA_R[1]	PIN_C13	VGA Red[1]	3.3V
VGA_R[2]	PIN_E13	VGA Red[2]	3.3V
VGA_R[3]	PIN_B12	VGA Red[3]	3.3V
VGA_R[4]	PIN_C12	VGA Red[4]	3.3V
VGA_R[5]	PIN_D12	VGA Red[5]	3.3V
VGA_R[6]	PIN_E12	VGA Red[6]	3.3V
VGA_R[7]	PIN_F13	VGA Red[7]	3.3V
VGA_G[0]	PIN_J9	VGA Green[0]	3.3V
VGA_G[1]	PIN_J10	VGA Green[1]	3.3V
VGA_G[2]	PIN_H12	VGA Green[2]	3.3V
VGA_G[3]	PIN_G10	VGA Green[3]	3.3V
VGA_G[4]	PIN_G11	VGA Green[4]	3.3V
VGA_G[5]	PIN_G12	VGA Green[5]	3.3V
VGA_G[6]	PIN_F11	VGA Green[6]	3.3V
VGA_G[7]	PIN_E11	VGA Green[7]	3.3V
VGA_B[0]	PIN_B13	VGA Blue[0]	3.3V
VGA_B[1]	PIN_G13	VGA Blue[1]	3.3V
VGA_B[2]	PIN_H13	VGA Blue[2]	3.3V
VGA_B[3]	PIN_F14	VGA Blue[3]	3.3V
VGA_B[4]	PIN_H14	VGA Blue[4]	3.3V
VGA_B[5]	PIN_F15	VGA Blue[5]	3.3V
VGA_B[6]	PIN_G15	VGA Blue[6]	3.3V
VGA_B[7]	PIN_J14	VGA Blue[7]	3.3V
VGA_CLK	PIN_A11	VGA Clock	3.3V
VGA_BLANK_N	PIN_F10	VGA BLANK	3.3V
VGA_HS	PIN_B11	VGA H_SYNC	3.3V
VGA_VS	PIN_D11	VGA V_SYNC	3.3V
VGA_SYNC_N	PIN_C10	VGA SYNC	3.3V

3.6.7 TV Decoder

The DE1-SoC board is equipped with an Analog Device ADV7180 TV decoder chip. The ADV7180 is an integrated video decoder that automatically detects and converts a standard analog baseband television signals (NTSC, PAL, and SECAM) into 4:2:2 component video data compatible with the 8-bit ITU-R BT.656 interface standard. The ADV7180 is compatible with a broad range of video devices, including DVD players, tape-based sources, broadcast sources, and security/surveillance cameras.

The registers in the TV decoder can be programmed by a serial I2C bus, which is connected to HPS or Cyclone V SoC FPGA through an I2C multiplexer as indicated in **Figure 3-23**. Note that the I2C address W/R of the TV decoder (U4) is 0x40/0x41. The pin assignments are listed in **Table 3-17**. Detailed information of the ADV7180 is available on the manufacturer's website, or in the DE1_SOC_datasheets\Video Decoder folder on the DE1-SoC System CD.

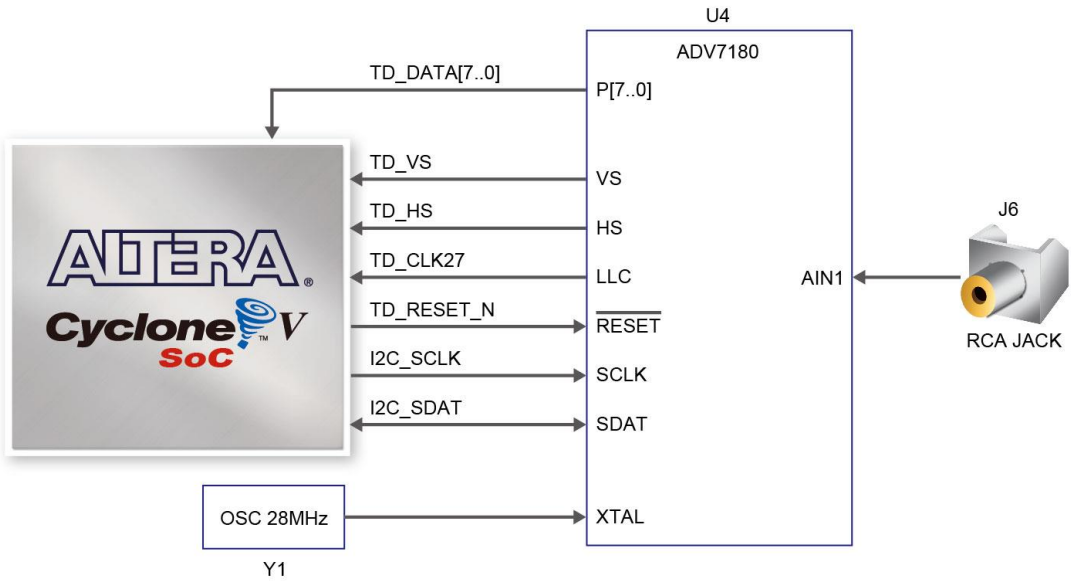


Figure 3-23 Connections between FPGA and TV Decoder

Table 3-17 TV Decoder Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
TD_DATA [0]	PIN_D2	TV Decoder Data[0]	3.3V
TD_DATA [1]	PIN_B1	TV Decoder Data[1]	3.3V
TD_DATA [2]	PIN_E2	TV Decoder Data[2]	3.3V
TD_DATA [3]	PIN_B2	TV Decoder Data[3]	3.3V
TD_DATA [4]	PIN_D1	TV Decoder Data[4]	3.3V
TD_DATA [5]	PIN_E1	TV Decoder Data[5]	3.3V
TD_DATA [6]	PIN_C2	TV Decoder Data[6]	3.3V
TD_DATA [7]	PIN_B3	TV Decoder Data[7]	3.3V
TD_HS	PIN_A5	TV Decoder H_SYNC	3.3V
TD_VS	PIN_A3	TV Decoder V_SYNC	3.3V
TD_CLK27	PIN_H15	TV Decoder Clock Input.	3.3V
TD_RESET_N	PIN_F6	TV Decoder Reset	3.3V
I2C_SCLK	PIN_J12 or PIN_E23	I2C Clock	3.3V
I2C_SDAT	PIN_K12 or PIN_C24	I2C Data	3.3V

3.6.8 IR Receiver

The board provides an infrared remote-control receiver module (model: IRM-V538/TR1), whose datasheet is offered in the Datasheets\ IR Receiver and Emitter folder on DE1-SoC System CD. The accompanied remote controller with an encoding chip of uPD6121G is very suitable of generating expected infrared signals. Figure 3-24 shows the related schematic of the IR receiver. Table 3-18 shows the IR receiver interface pin assignments.

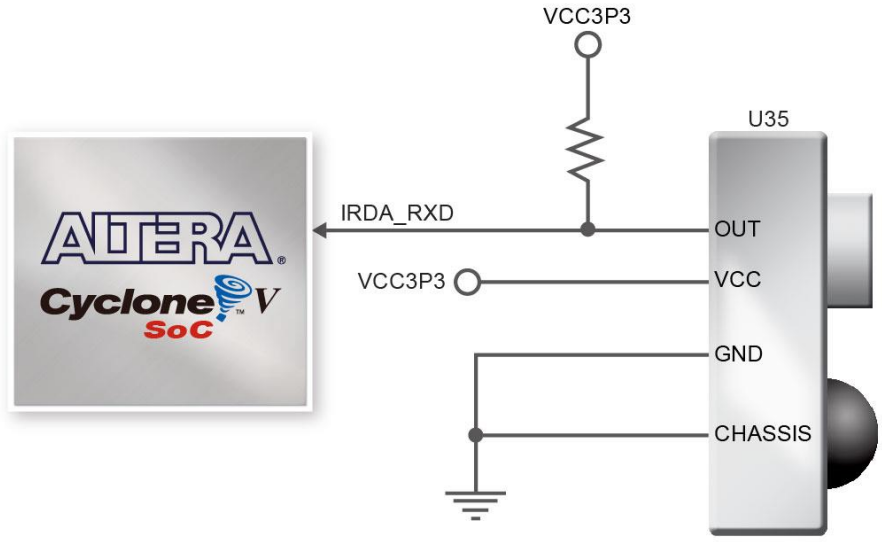


Figure 3-24 Connection between FPGA and IR Receiver

Table 3-18 Pin Assignments for IR

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
IRDA_RXD	PIN_ AA30	IR Receiver	3.3V

3.6.9 IR Emitter LED

The board provides an IR Emitter LED for IR communication which is widely used for operating the television device wirelessly from a short line-of-sight distance. Match this IR Emitter LED with an IR receiver will allow the board to communicate with similarly equipped system. Figure 3-25 shows the related schematic of the IR emitter LED. Table 3-19 shows the IR emitter interface pin assignments.

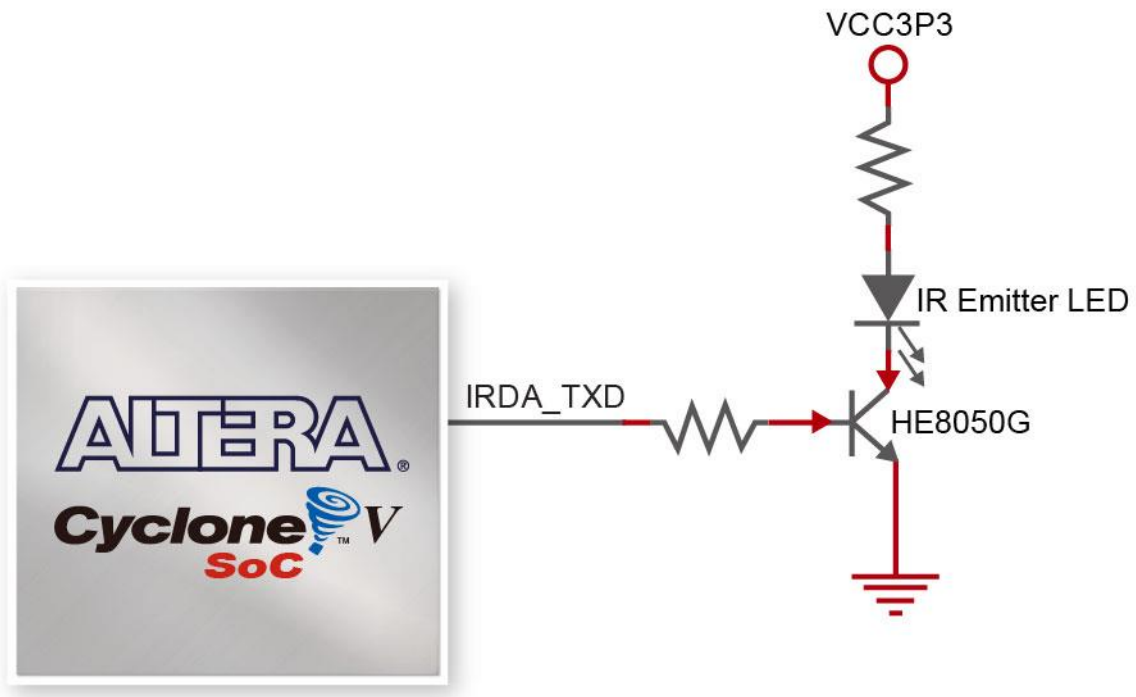


Figure 3-25 Connection between FPGA and IR Emitter LED

Table 3-19 Pin Assignments for IR

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
IRDA_TXD	PIN_ AB30	IR Emitter	3.3V

3.6.10 SDRAM Memory on FPGA

The board features 64MB of SDRAM, implemented using a 64MB (32Mx16) SDRAM device. The device consists of 16-bit data line, control line and address line connected to the FPGA. This chip use the 3.3V LVCMOS signaling standard. Connections between FPGA and SDRAM are shown in Figure 3-26, while the pin assignments are listed in Table 3-20.

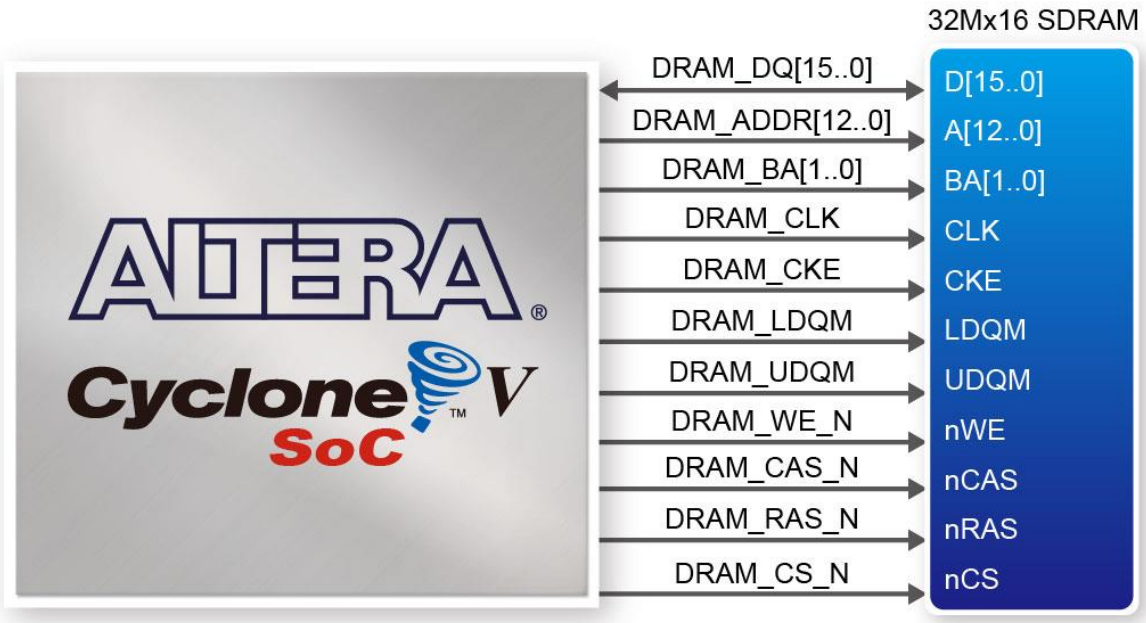


Figure 3-26 Connections between FPGA and SDRAM

Table 3-20 SDRAM Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
DRAM_ADDR[0]	PIN_AK14	SDRAM Address[0]	3.3V
DRAM_ADDR[1]	PIN_AH14	SDRAM Address[1]	3.3V
DRAM_ADDR[2]	PIN_AG15	SDRAM Address[2]	3.3V
DRAM_ADDR[3]	PIN_AE14	SDRAM Address[3]	3.3V
DRAM_ADDR[4]	PIN_AB15	SDRAM Address[4]	3.3V
DRAM_ADDR[5]	PIN_AC14	SDRAM Address[5]	3.3V
DRAM_ADDR[6]	PIN_AD14	SDRAM Address[6]	3.3V
DRAM_ADDR[7]	PIN_AF15	SDRAM Address[7]	3.3V
DRAM_ADDR[8]	PIN_AH15	SDRAM Address[8]	3.3V
DRAM_ADDR[9]	PIN_AG13	SDRAM Address[9]	3.3V
DRAM_ADDR[10]	PIN_AG12	SDRAM Address[10]	3.3V
DRAM_ADDR[11]	PIN_AH13	SDRAM Address[11]	3.3V
DRAM_ADDR[12]	PIN_AJ14	SDRAM Address[12]	3.3V
DRAM_DQ[0]	PIN_AK6	SDRAM Data[0]	3.3V
DRAM_DQ[1]	PIN_AJ7	SDRAM Data[1]	3.3V
DRAM_DQ[2]	PIN_AK7	SDRAM Data[2]	3.3V
DRAM_DQ[3]	PIN_AK8	SDRAM Data[3]	3.3V
DRAM_DQ[4]	PIN_AK9	SDRAM Data[4]	3.3V
DRAM_DQ[5]	PIN_AG10	SDRAM Data[5]	3.3V
DRAM_DQ[6]	PIN_AK11	SDRAM Data[6]	3.3V

DRAM_DQ[7]	PIN_AJ11	SDRAM Data[7]	3.3V
DRAM_DQ[8]	PIN_AH10	SDRAM Data[8]	3.3V
DRAM_DQ[9]	PIN_AJ10	SDRAM Data[9]	3.3V
DRAM_DQ[10]	PIN_AJ9	SDRAM Data[10]	3.3V
DRAM_DQ[11]	PIN_AH9	SDRAM Data[11]	3.3V
DRAM_DQ[12]	PIN_AH8	SDRAM Data[12]	3.3V
DRAM_DQ[13]	PIN_AH7	SDRAM Data[13]	3.3V
DRAM_DQ[14]	PIN_AJ6	SDRAM Data[14]	3.3V
DRAM_DQ[15]	PIN_AJ5	SDRAM Data[15]	3.3V
DRAM_BA[0]	PIN_AF13	SDRAM Bank Address[0]	3.3V
DRAM_BA[1]	PIN_AJ12	SDRAM Bank Address[1]	3.3V
DRAM_LDQM	PIN_AB13	SDRAM byte Data Mask[0]	3.3V
DRAM_UDQM	PIN_AK12	SDRAM byte Data Mask[1]	3.3V
DRAM_RAS_N	PIN_AE13	SDRAM Row Address Strobe	3.3V
DRAM_CAS_N	PIN_AF11	SDRAM Column Address Strobe	3.3V
DRAM_CKE	PIN_AK13	SDRAM Clock Enable	3.3V
DRAM_CLK	PIN_AH12	SDRAM Clock	3.3V
DRAM_WE_N	PIN_AA13	SDRAM Write Enable	3.3V
DRAM_CS_N	PIN_AG11	SDRAM Chip Select	3.3V

3.6.11 PS/2 Serial Port

The DE1-SoC board includes a standard PS/2 interface and a connector for a PS/2 keyboard or mouse. **Figure 3-27** shows the schematic of the PS/2 circuit. In addition, users can use the PS/2 keyboard and mouse on the DE1-SoC board simultaneously by plugging an extension PS/2 Y-Cable (See **Figure 3-28**). Instructions for using a PS/2 mouse or keyboard can be found by performing an appropriate search on various educational websites. The pin assignments for the associated interface are shown in **Table 3-21.**

Note: If users connect only one PS/2 equipment, the PS/2 interface between FPGA I/O should be "PS2_CLK" and "PS2_DAT".

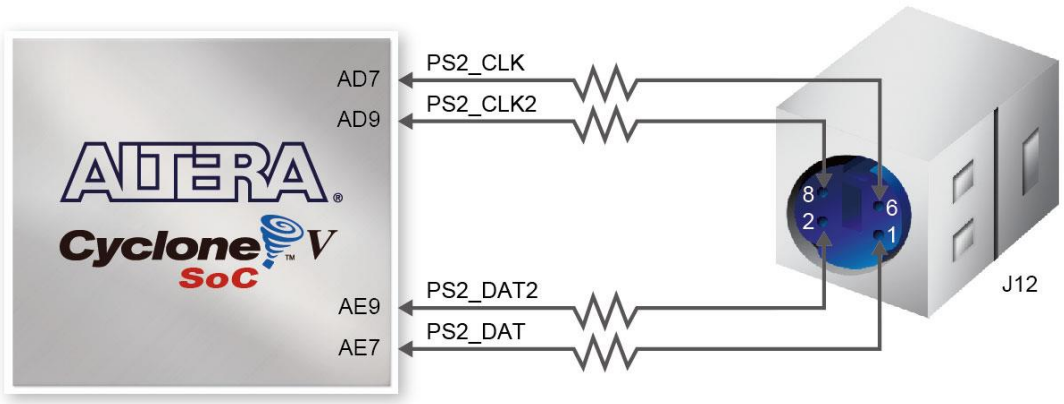


Figure 3-27 Connection between FPGA and PS/2

Figure 3-28 Y-Cable use for both Keyboard and Mouse

Table 3-21 PS/2 Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
PS2_CLK	PIN_AD7	PS/2 Clock	3.3V
PS2_DAT	PIN_AE7	PS/2 Data	3.3V
PS2_CLK2	PIN_AD9	PS/2 Clock (reserved for second PS/2 device)	3.3V
PS2_DAT2	PIN_AE9	PS/2 Data (reserved for second PS/2 device)	3.3V

3.6.12 A/D Converter and 2x5 Header

The DE1-SoC contains AD7928, a lower power, eight-channel CMOS 12-bit analog-to-digital converter. This A-to-D provides conversion throughput rates up to 1MSPS. The analog input range for all input channels can extend from 0 V to 2.5 V or 0 V to 5V as selected via the RANGE bit in the control register. It can be configured to accept eight input signals at inputs ADC_IN0 through ADC_IN7. These eight input signals are connected to the 2x5 header, as shown in **Figure 3-29**.

For more detailed information on the A/D converter chip, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD.

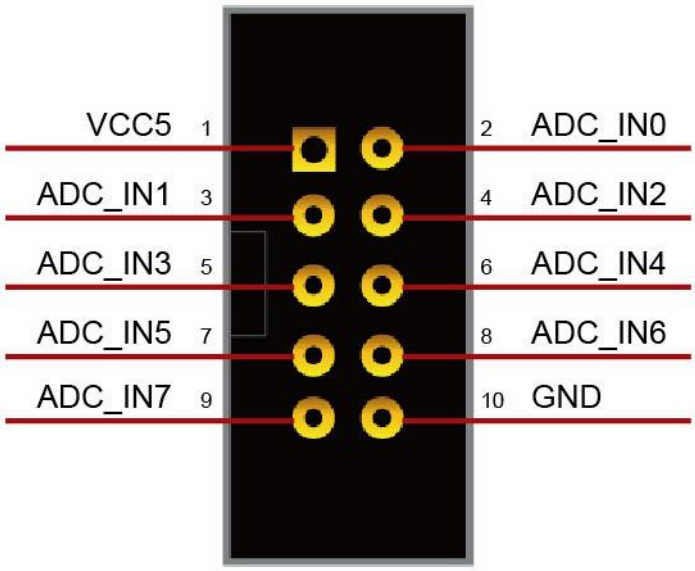


Figure 3-29 Pin distribution of the 2x5 Header

Figure 3-30 shows the connections on the 2x5 header, A/D converter and Cyclone V SoC device.

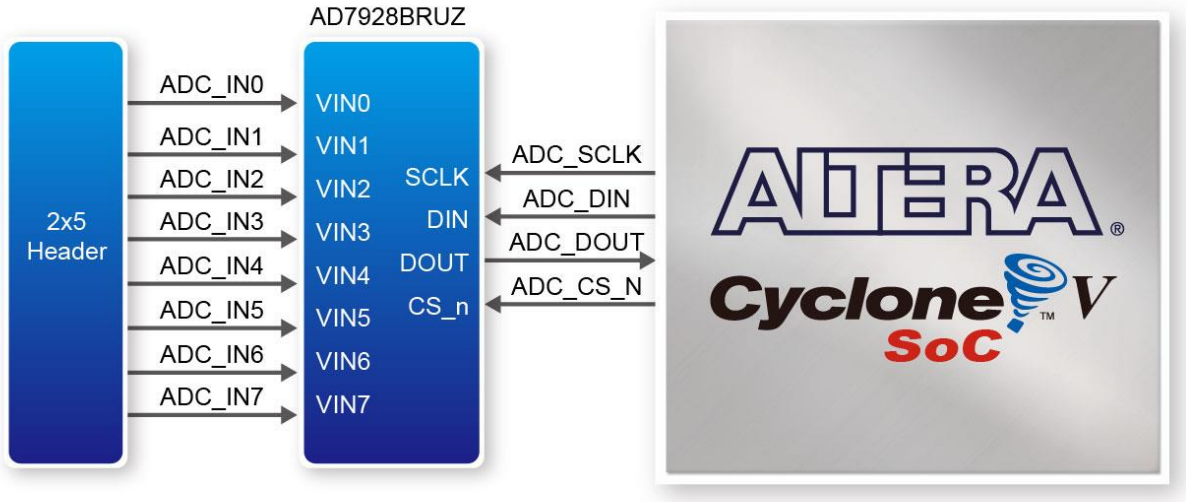


Figure 3-30 Wiring for 2x5 header and A/D converter

Table 3-22 Pin Assignments for ADC

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
ADC_CS_N	PIN_AJ4	Chip select	3.3V
ADC_DOUT	PIN_AK3	Digital data input	3.3V
ADC_DIN	PIN_AK4	Digital data output	3.3V
ADC_SCLK	PIN_AK2	Digital clock input	3.3V

3.7 Interface on Hard Processor System (HPS)

This section introduces the interfaces connected to the HPS section of the FPGA. Users can access these interfaces via the HPS processor.

3.7.1 User Push-button and LED on HPS

Like the FPGA, the HPS also features its own set of switches, buttons, LEDs, and other user interfaces. Users can control these interfaces for observing HPS status and debugging.

Table 3-23 gives the all the pin assignments of all the user interfaces.

Table 3-23 Pin Assignments for LEDs, Switches and Buttons

Signal Name	HPS GPIO	Register/bit	Function
HPS_KEY	GPIO54	GPIO1[25]	I/O
HPS_LED	GPIO53	GPIO1[24]	I/O

3.7.2 Gigabit Ethernet

The board provides Ethernet support via an external Micrel KSZ9021RN PHY chip and HPS Ethernet MAC function. The KSZ9021RN chip with integrated 10/100/1000 Mbps Gigabit Ethernet transceiver support RGMII MAC interfaces. **Figure 3-31** shows the connection setup between the Gigabit Ethernet PHY and Cyclone V SoC FPGA.

The associated pin assignments are listed in **Table 3-24**. For detailed information on how to use the KSZ9021RN refers to its datasheet and application notes, which are available on the manufacturer's website.

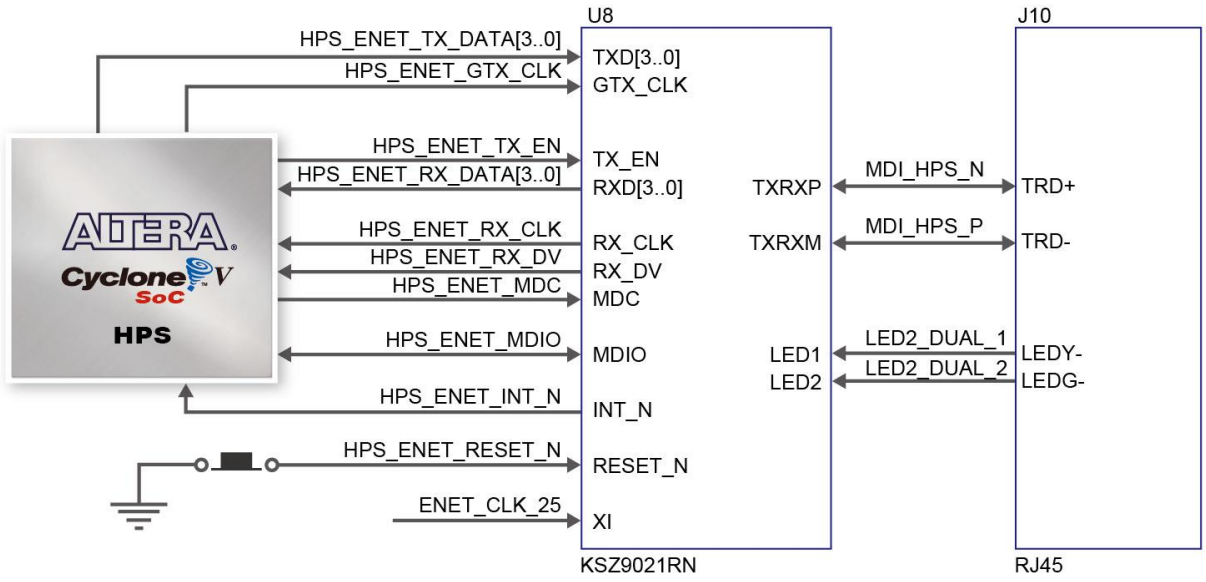


Figure 3-31 Connections between Cyclone V SoC FPGA and Ethernet

Table 3-24 Pin Assignments for Ethernet PHY

Signal Name	FPGA Pin No.	Description	I/O Standard
HPS_ENET_TX_EN	PIN_A20	GMII and MII transmit enable	3.3V
HPS_ENET_TX_DATA[0]	PIN_F20	MII transmit data[0]	3.3V

HPS_ENET_TX_DATA[1]	PIN_J19	MII transmit data[1]	3.3V
HPS_ENET_TX_DATA[2]	PIN_F21	MII transmit data[2]	3.3V
HPS_ENET_TX_DATA[3]	PIN_F19	MII transmit data[3]	3.3V
HPS_ENET_RX_DV	PIN_K17	GMII and MII receive data valid	3.3V
HPS_ENET_RX_DATA[0]	PIN_A21	GMII and MII receive data[0]	3.3V
HPS_ENET_RX_DATA[1]	PIN_B20	GMII and MII receive data[1]	3.3V
HPS_ENET_RX_DATA[2]	PIN_B18	GMII and MII receive data[2]	3.3V
HPS_ENET_RX_DATA[3]	PIN_D21	GMII and MII receive data[3]	3.3V
HPS_ENET_RX_CLK	PIN_G20	GMII and MII receive clock	3.3V
HPS_ENET_RESET_N	PIN_E18	Hardware Reset Signal	3.3V
HPS_ENET_MDIO	PIN_E21	Management Data	3.3V
HPS_ENET_MDC	PIN_B21	Management Data Clock Reference	3.3V
HPS_ENET_INT_N	PIN_C19	Interrupt Open Drain Output	3.3V
HPS_ENET_GTX_CLK	PIN_H19	GMII Transmit Clock	3.3V

Additionally, the Ethernet PHY (KSZ9021RNI) LED status has been set to two LED mode. The LED control signals are connected to LEDs (yellow and green) on the RJ45 connector. States and definitions can be found in **Table 3-25,** which can display the current status of the Ethernet. For example once the green LED lights on, the board has been connected to Gigabit Ethernet.

Table 3-25 LED Mode-Pin Definition

LED (State)		*LED (Definition)*		*Link /Activity*
LEDG	*LEDY*	*LEDG*	*LEDY*	
H	H	OFF	OFF	Link off
L	H	ON	OFF	1000 Link / No Activity
Toggle	H	Blinking	OFF	1000 Link / Activity (RX, TX)
H	L	OFF	ON	100 Link / No Activity
H	Toggle	OFF	Blinking	100 Link / Activity (RX, TX)
L	L	ON	ON	10 Link/ No Activity
Toggle	Toggle	Blinking	Blinking	10 Link / Activity (RX, TX)

3.7.3 UART

The board has one UART interface connected for communication with the HPS. This interface wouldn't support HW flow control signals. The physical interface is done using UART-USB onboard bridge from an FT232R chip and connects to the host using an USB Mini-B connector. For detailed information on how to use the transceiver, please refer to the datasheet, which is available on the manufacturer's website, or in the Datasheets\UART TO USB folder on the DE1-SoC System CD. **Figure 3-32** shows the related schematics, and **Table 3-26** lists the pin assignments of HPS in Cyclone V SoC FPGA.

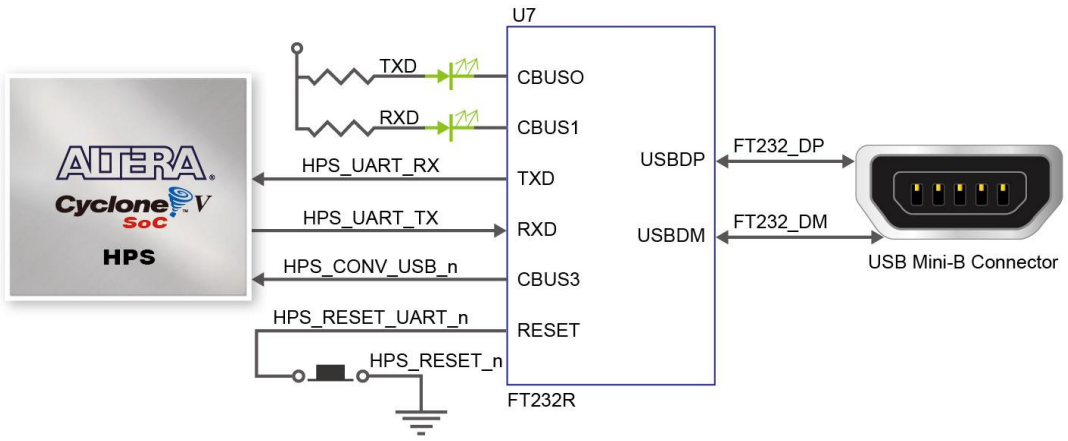


Figure 3-32 Connections between the Cyclone V SoC FPGA and FT232R Chip

Table 3-26 UART Interface I/O

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HPS_UART_RX	PIN_B25	HPS UART Receiver	3.3V
HPS_UART_TX	PIN_C25	HPS UART Transmitter	3.3V
HPS_CONV_USB_N	PIN_B15	Reserve	3.3V

3.7.4 DDR3 Memory on HPS

The DDR3 devices that are connected to the HPS are the exact same devices connected to the FPGA in capacity (1GB) and data-width (32-bit), comprised of two x16 devices with a single address/command bus. This interface connects to dedicated Hard Memory Controller for HPS I/O banks and the target speed is 400 MHz. Table 3-27 lists DDR3 pin assignments, I/O standards and descriptions with Cyclone V SoC FPGA.

Table 3-27 Pin Assignments for DDR3 Memory

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HPS_DDR3_A[0]	PIN_F26	HPS DDR3 Address[0]	SSTL-15 Class I
HPS_DDR3_A[1]	PIN_G30	HPS DDR3 Address[1]	SSTL-15 Class I
HPS_DDR3_A[2]	PIN_F28	HPS DDR3 Address[2]	SSTL-15 Class I
HPS_DDR3_A[3]	PIN_F30	HPS DDR3 Address[3]	SSTL-15 Class I
HPS_DDR3_A[4]	PIN_J25	HPS DDR3 Address[4]	SSTL-15 Class I
HPS_DDR3_A[5]	PIN_J27	HPS DDR3 Address[5]	SSTL-15 Class I
HPS_DDR3_A[6]	PIN_F29	HPS DDR3 Address[6]	SSTL-15 Class I
HPS_DDR3_A[7]	PIN_E28	HPS DDR3 Address[7]	SSTL-15 Class I
HPS_DDR3_A[8]	PIN_H27	HPS DDR3 Address[8]	SSTL-15 Class I
HPS_DDR3_A[9]	PIN_G26	HPS DDR3 Address[9]	SSTL-15 Class I
HPS_DDR3_A[10]	PIN_D29	HPS DDR3 Address[10]	SSTL-15 Class I

HPS_DDR3_A[11]	PIN_C30	HPS DDR3 Address[11]	SSTL-15 Class I
HPS_DDR3_A[12]	PIN_B30	HPS DDR3 Address[12]	SSTL-15 Class I
HPS_DDR3_A[13]	PIN_C29	HPS DDR3 Address[13]	SSTL-15 Class I
HPS_DDR3_A[14]	PIN_H25	HPS DDR3 Address[14]	SSTL-15 Class I
HPS_DDR3_BA[0]	PIN_E29	HPS DDR3 Bank Address[0]	SSTL-15 Class I
HPS_DDR3_BA[1]	PIN_J24	HPS DDR3 Bank Address[1]	SSTL-15 Class I
HPS_DDR3_BA[2]	PIN_J23	HPS DDR3 Bank Address[2]	SSTL-15 Class I
HPS_DDR3_CAS_n	PIN_E27	DDR3 Column Address Strobe	SSTL-15 Class I
HPS_DDR3_CKE	PIN_L29	HPS DDR3 Clock Enable	SSTL-15 Class I
HPS_DDR3_CK_n	PIN_L23	HPS DDR3 Clock	Differential 1.5-V SSTL Class I
HPS_DDR3_CK_p	PIN_M23	HPS DDR3 Clock p	Differential 1.5-V SSTL Class I
HPS_DDR3_CS_n	PIN_H24	HPS DDR3 Chip Select	SSTL-15 Class I
HPS_DDR3_DM[0]	PIN_K28	HPS DDR3 Data Mask[0]	SSTL-15 Class I
HPS_DDR3_DM[1]	PIN_M28	HPS DDR3 Data Mask[1]	SSTL-15 Class I
HPS_DDR3_DM[2]	PIN_R28	HPS DDR3 Data Mask[2]	SSTL-15 Class I
HPS_DDR3_DM[3]	PIN_W30	HPS DDR3 Data Mask[3]	SSTL-15 Class I
HPS_DDR3_DQ[0]	PIN_K23	HPS DDR3 Data[0]	SSTL-15 Class I
HPS_DDR3_DQ[1]	PIN_K22	HPS DDR3 Data[1]	SSTL-15 Class I
HPS_DDR3_DQ[2]	PIN_H30	HPS DDR3 Data[2]	SSTL-15 Class I
HPS_DDR3_DQ[3]	PIN_G28	HPS DDR3 Data[3]	SSTL-15 Class I
HPS_DDR3_DQ[4]	PIN_L25	HPS DDR3 Data[4]	SSTL-15 Class I
HPS_DDR3_DQ[5]	PIN_L24	HPS DDR3 Data[5]	SSTL-15 Class I
HPS_DDR3_DQ[6]	PIN_J30	HPS DDR3 Data[6]	SSTL-15 Class I
HPS_DDR3_DQ[7]	PIN_J29	HPS DDR3 Data[7]	SSTL-15 Class I
HPS_DDR3_DQ[8]	PIN_K26	HPS DDR3 Data[8]	SSTL-15 Class I
HPS_DDR3_DQ[9]	PIN_L26	HPS DDR3 Data[9]	SSTL-15 Class I
HPS_DDR3_DQ[10]	PIN_K29	HPS DDR3 Data[10]	SSTL-15 Class I
HPS_DDR3_DQ[11]	PIN_K27	HPS DDR3 Data[11]	SSTL-15 Class I
HPS_DDR3_DQ[12]	PIN_M26	HPS DDR3 Data[12]	SSTL-15 Class I
HPS_DDR3_DQ[13]	PIN_M27	HPS DDR3 Data[13]	SSTL-15 Class I
HPS_DDR3_DQ[14]	PIN_L28	HPS DDR3 Data[14]	SSTL-15 Class I
HPS_DDR3_DQ[15]	PIN_M30	HPS DDR3 Data[15]	SSTL-15 Class I
HPS_DDR3_DQ[16]	PIN_U26	HPS DDR3 Data[16]	SSTL-15 Class I
HPS_DDR3_DQ[17]	PIN_T26	HPS DDR3 Data[17]	SSTL-15 Class I
HPS_DDR3_DQ[18]	PIN_N29	HPS DDR3 Data[18]	SSTL-15 Class I
HPS_DDR3_DQ[19]	PIN_N28	HPS DDR3 Data[19]	SSTL-15 Class I
HPS_DDR3_DQ[20]	PIN_P26	HPS DDR3 Data[20]	SSTL-15 Class I
HPS_DDR3_DQ[21]	PIN_P27	HPS DDR3 Data[21]	SSTL-15 Class I
HPS_DDR3_DQ[22]	PIN_N27	HPS DDR3 Data[22]	SSTL-15 Class I
HPS_DDR3_DQ[23]	PIN_R29	HPS DDR3 Data[23]	SSTL-15 Class I
HPS_DDR3_DQ[24]	PIN_P24	HPS DDR3 Data[24]	SSTL-15 Class I
HPS_DDR3_DQ[25]	PIN_P25	HPS DDR3 Data[25]	SSTL-15 Class I
HPS_DDR3_DQ[26]	PIN_T29	HPS DDR3 Data[26]	SSTL-15 Class I
HPS_DDR3_DQ[27]	PIN_T28	HPS DDR3 Data[27]	SSTL-15 Class I

HPS_DDR3_DQ[28]	PIN_R27	HPS DDR3 Data[28]	SSTL-15 Class I
HPS_DDR3_DQ[29]	PIN_R26	HPS DDR3 Data[29]	SSTL-15 Class I
HPS_DDR3_DQ[30]	PIN_V30	HPS DDR3 Data[30]	SSTL-15 Class I
HPS_DDR3_DQ[31]	PIN_W29	HPS DDR3 Data[31]	SSTL-15 Class I
HPS_DDR3_DQS_n[0]	PIN_M19	HPS DDR3 Data Strobe n[0]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_n[1]	PIN_N24	HPS DDR3 Data Strobe n[1]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_n[2]	PIN_R18	HPS DDR3 Data Strobe n[2]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_n[3]	PIN_R21	HPS DDR3 Data Strobe n[3]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[0]	PIN_N18	HPS DDR3 Data Strobe p[0]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[1]	PIN_N25	HPS DDR3 Data Strobe p[1]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[2]	PIN_R19	HPS DDR3 Data Strobe p[2]	Differential 1.5-V SSTL Class I
HPS_DDR3_DQS_p[3]	PIN_R22	HPS DDR3 Data Strobe p[3]	Differential 1.5-V SSTL Class I
HPS_DDR3_ODT	PIN_H28	HPS DDR3 On-die Termination	SSTL-15 Class I
HPS_DDR3_RAS_n	PIN_D30	DDR3 Row Address Strobe	SSTL-15 Class I
HPS_DDR3_RESET_n	PIN_P30	HPS DDR3 Reset	SSTL-15 Class I
HPS_DDR3_WE_n	PIN_C28	HPS DDR3 Write Enable	SSTL-15 Class I
HPS_DDR3_RZQ	PIN_D27	External reference ball for output drive calibration	1.5 V

3.7.5 Micro SD

The board supports Micro SD card interface using x4 data lines. And it may contain secondary boot code for HPS. Figure 3-33 shows the related signals.

Finally, Table 3-28 lists all the associated pins for interfacing HPS respectively.

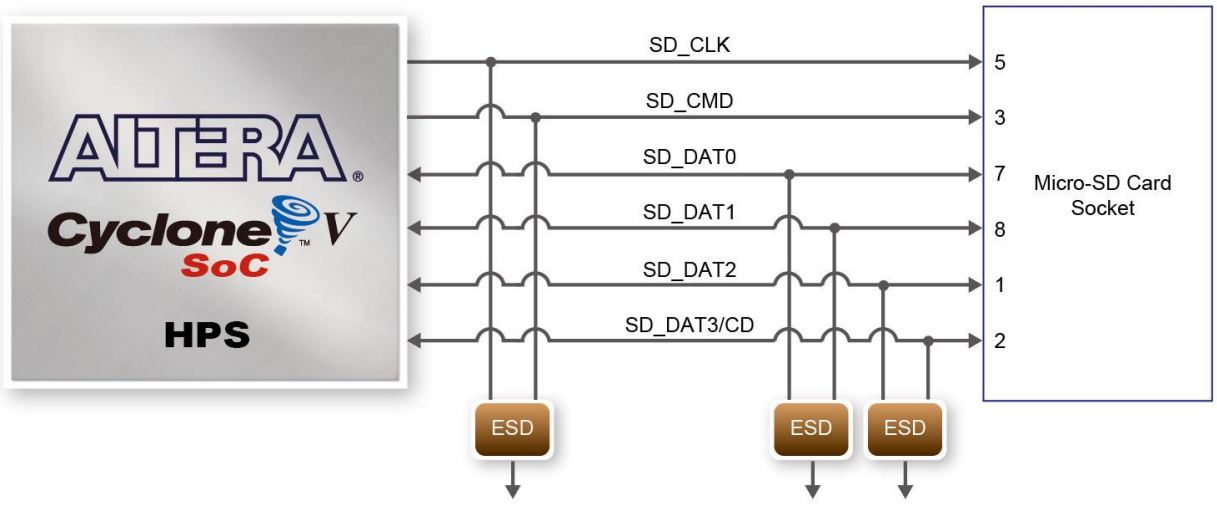


Figure 3-33 Connections between Cyclone V SoC FPGA and SD Card Socket

Table 3-28 SD Card Socket Pin Assignments

Signal Name	FPGA Pin No.	Description	I/O Standard
HPS_SD_CLK	PIN_A16	HPS SD Clock	3.3V
HPS_SD_CMD	PIN_F18	HPS SD Command Line	3.3V
HPS_SD_DATA[0]	PIN_G18	HPS SD Data[0]	3.3V
HPS_SD_DATA[1]	PIN_C17	HPS SD Data[1]	3.3V
HPS_SD_DATA[2]	PIN_D17	HPS SD Data[2]	3.3V
HPS_SD_DATA[3]	PIN_B16	HPS SD Data[3]	3.3V

3.7.6 2-port USB Host

The board provides 2-port USB 2.0 host interfaces using the SMSC USB3300 controller and 2-port hub controller. A SMSC USB3300 device in a 32-pin QFN package device is used to interface to a SMSC USB2512B. This device supports UTMI+ Low Pin Interface (ULPI) to communicate to USB 2.0 controller in HPS. By connecting the ID pin of USB3300 to ground, the PHY operates in Host mode. When operating in Host mode, the interface will supply the power to the device through the 2-port USB type-A interface. Figure 3-34 shows the schematic diagram of the USB circuitry; the pin assignments for the associated interface are listed in Table 3-29.

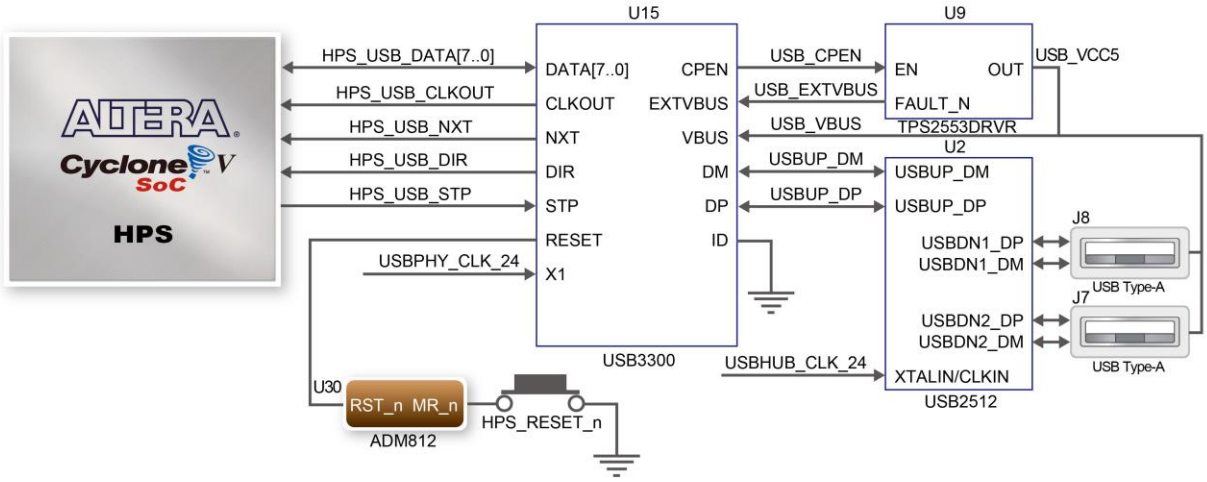


Figure 3-34 Connections between Cyclone V SoC FPGA and USB OTG PHY

Table 3-29 USB OTG PHY Pin Assignments

Signal Name	FPGA Pin No.	Description	I/O Standard
HPS_USB_CLKOUT	PIN_N16	60MHz Reference Clock Output	3.3V
HPS_USB_DATA[0]	PIN_E16	HPS USB_DATA[0]	3.3V

HPS_USB_DATA[1]	PIN_G16	HPS USB_DATA[1]	3.3V
HPS_USB_DATA[2]	PIN_D16	HPS USB_DATA[2]	3.3V
HPS_USB_DATA[3]	PIN_D14	HPS USB_DATA[3]	3.3V
HPS_USB_DATA[4]	PIN_A15	HPS USB_DATA[4]	3.3V
HPS_USB_DATA[5]	PIN_C14	HPS USB_DATA[5]	3.3V
HPS_USB_DATA[6]	PIN_D15	HPS USB_DATA[6]	3.3V
HPS_USB_DATA[7]	PIN_M17	HPS USB_DATA[7]	3.3V
HPS_USB_DIR	PIN_E14	Direction of the Data Bus	3.3V
HPS_USB_NXT	PIN_A14	Throttle the Data	3.3V
HPS_USB_RESET	PIN_G17	HPS USB PHY Reset	3.3V
HPS_USB_STP	PIN_C15	Stop Data Stream on the Bus	3.3V

3.7.7 G-Sensor

The board is equipped with a digital accelerometer sensor module. The ADXL345 is a small, thin, ultralow power assumption 3-axis accelerometer with high-resolution measurement. Digitalized output is formatted as 16-bit twos complement and can be accessed using I2C interface. The I2C address of the G-Sensor device is 0xA6/0xA7. For more detailed information of better using this chip, please refer to its datasheet which is available on manufacturer's website or under the Datasheet folder of the DE1-SoC System CD. **Figure 3-35** shows the connections between ADXL345 and HPS. The associated pin assignments are listed in **Table 3-30**.

Figure 3-35 Connections between Cyclone V SoC FPGA and G-Sensor

Table 3-30 G-Sensor Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HPS_GSENSOR_INT	PIN_B22	HPS GSENSOR Interrupt Output	3.3V

HPS_I2C1_SCLK	PIN_E23	HPS I2C Clock (share bus with LTC)	3.3V
HPS_I2C1_SDAT	PIN_C24	HPS I2C Data (share bus)	3.3V

3.7.8 LTC Connector

The board allows connection to interface card from Linear Technology. The interface is implemented using a14-pin header that can be connected to a variety of demo boards from Linear Technology. It will be connected to SPI Master and I2C ports of the HPS to allow bidirectional communication with two types of protocols. The 14-pin header will allow for GPIO, SPI and I2C extension for user purposes if the interfaces to Linear Technology board aren't in use. Connections between the LTC connector and the HPS are shown in **Figure 3-36,** and the functions of the 14 pins is listed in **Table 3-31**.

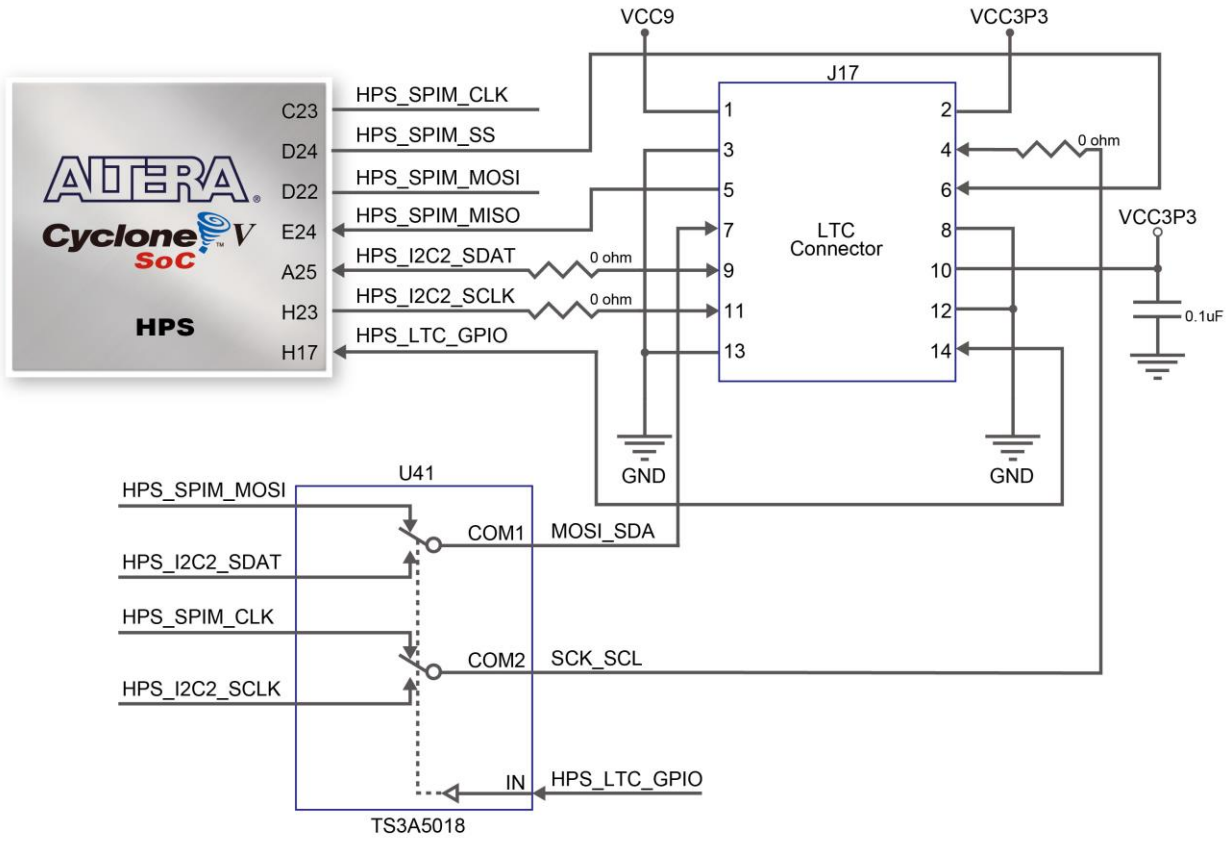


Figure 3-36 Connections between the LTC Connector and HPS

Table 3-31 LTC Connector Pin Assignments

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HPS_LTC_GPIO	PIN_H17	HPS LTC GPIO	3.3V
HPS_I2C2_SCLK	PIN_H23	HPS I2C2 Clock (share bus with	3.3V

		G-Sensor)	
HPS_I2C2_SDAT	PIN_A25	HPS I2C2 Data (share bus with G-Sensor)	3.3V
HPS_SPIM_CLK	PIN_C23	SPI Clock	3.3V
HPS_SPIM_MISO	PIN_E24	SPI Master Input/Slave Output	3.3V
HPS_SPIM_MOSI	PIN_D22	SPI Master Output /Slave Input	3.3V
HPS_SPIM_SS	PIN_D24	SPI Slave Select	3.3V

Chapter 4

DE1-SoC System Builder

This chapter describes how users can create a custom design project on the board by using the DE1-SoC Software Tool – DE1-SoC System Builder.

4.1 Introduction

The DE1-SoC System Builder is a Windows-based software utility, designed to assist users to create a Quartus II project for the board within minutes. The generated Quartus II project files include:

- Quartus II Project File (.qpf)
- Quartus II Setting File (.qsf)
- Top-Level Design File (.v)
- Synopsis Design Constraints file (.sdc)
- Pin Assignment Document (.htm)

By providing the above files, the DE1-SoC System Builder prevents occurrence of situations that are prone to errors when users manually edit the top-level design file or place pin assignments. The common mistakes that users encounter are the following:

1. Board damage due to wrong pin/bank voltage assignments.
2. Board malfunction caused by wrong device connections or missing pin counts for connected ends.
3. Performance degradation due to improper pin assignments.

4.2 General Design Flow

This section will introduce the general design flow to build a project for the development board via the DE1-SoC System Builder. The general design flow is illustrated in **Figure 4-1**.

Users should launch the DE1-SoC System Builder and create a new project according to their design requirements. When users complete the settings, the DE1-SoC System Builder will generate two major files, a top-level design file (.v) and a Quartus II setting file (.qsf).

The top-level design file contains top-level Verilog HDL wrapper for users to add their own design/logic. The Quartus II setting file contains information such as FPGA device type, top-level pin assignment, and the I/O standard for each user-defined I/O pin.

Finally, the Quartus II programmer must be used to download SOF file to the development board using a JTAG interface.

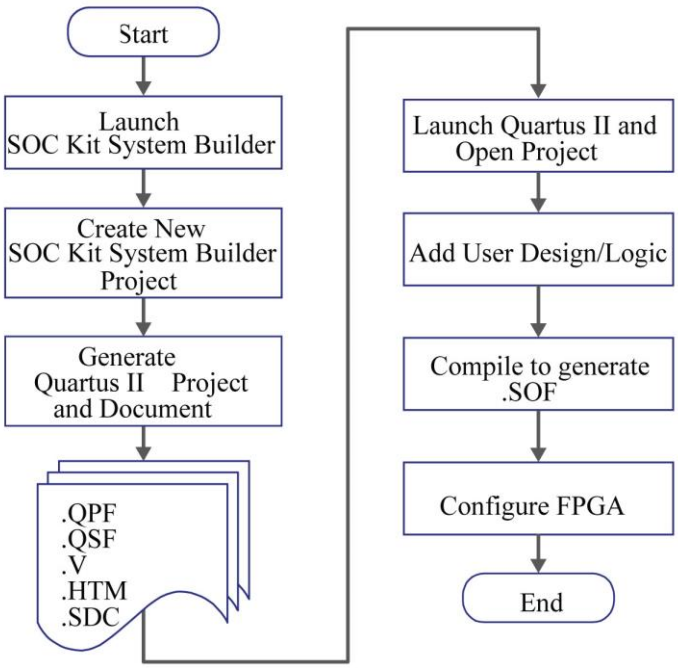


Figure 4-1 General design flow of building a design

4.3 Using DE1-SoC System Builder

This section provides the detailed procedures on how the DE1-SoC System Builder is used.

■ Install and launch the DE1-SoC System Builder

The DE1-SoC System Builder is located in the directory: "Tools\SystemBuilder" on the DE1-SoC

System CD. Users can copy the whole folder to a host computer without installing the utility. Launch the DE1-SoC System Builder by executing the DE1-SoC SystemBuilder.exe on the host computer and the GUI window will appear as shown in Figure 4-2.

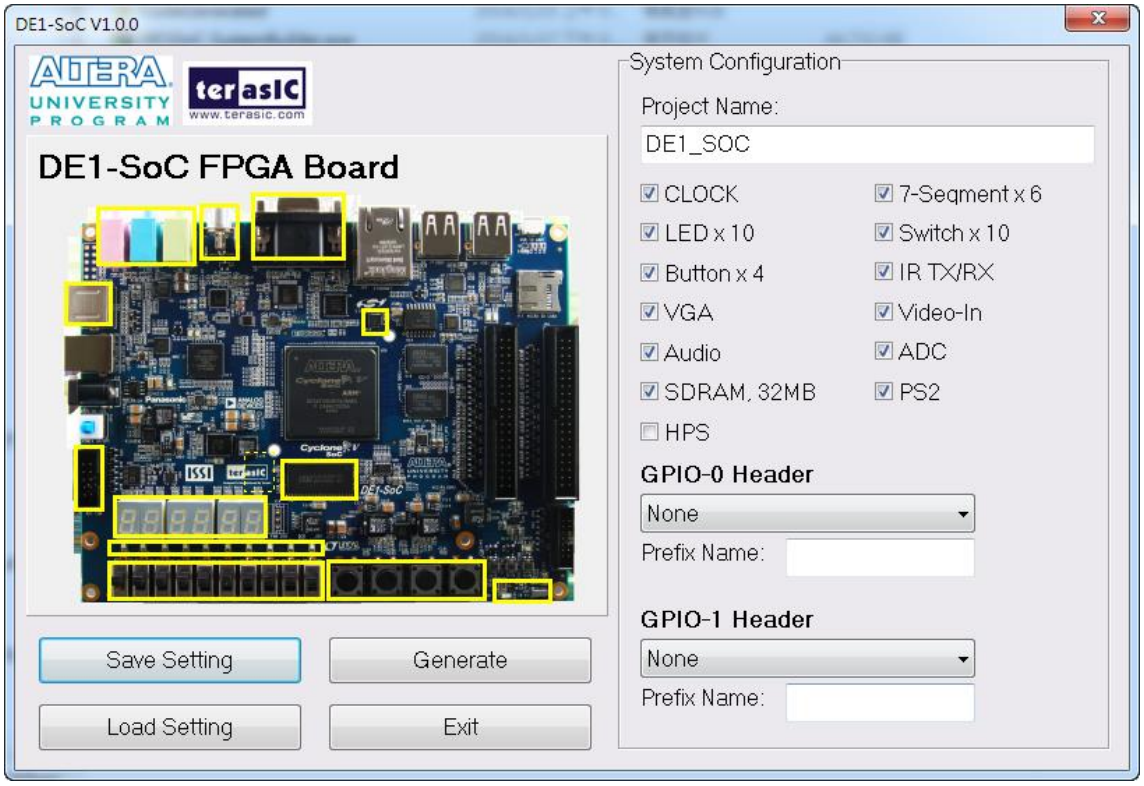


Figure 4-2 The DE1-SoC System Builder window

■ Input Project Name

Input project name as show in Figure 4-3.

Project Name: Type in an appropriate name here, it will automatically be assigned as the name of your top-level design entity.

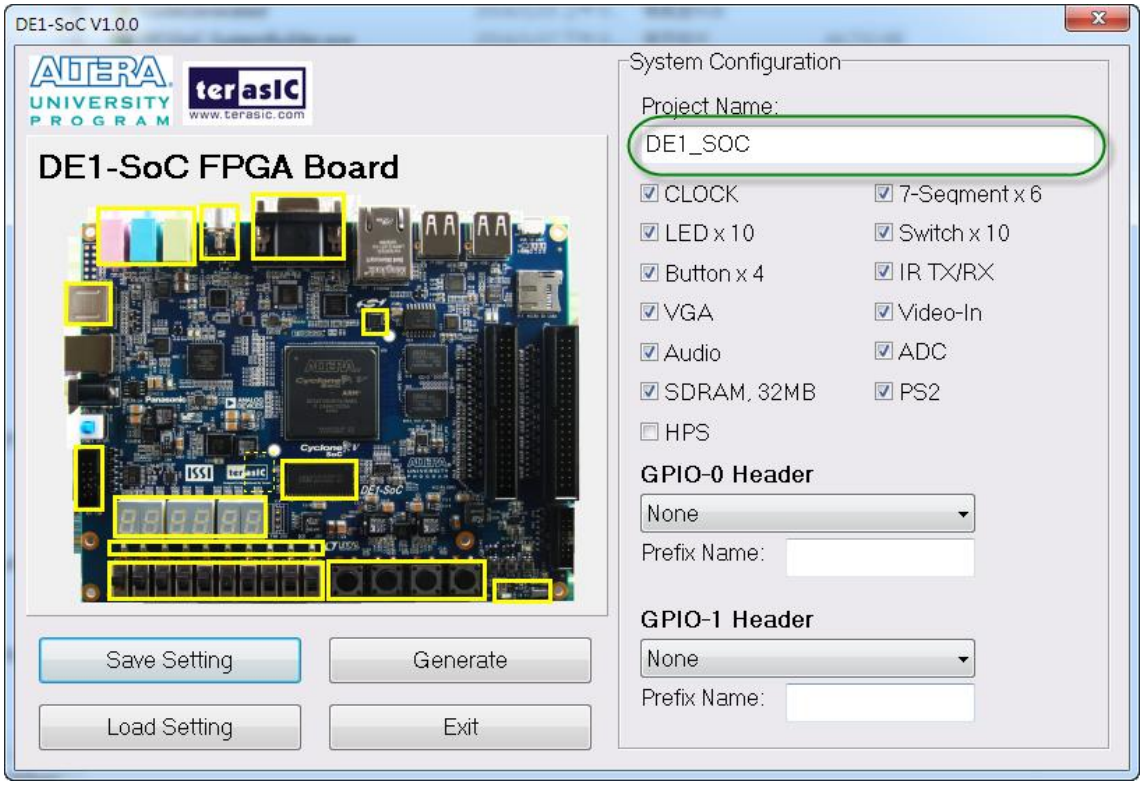


Figure 4-3 Board Type and Project Name

■ System Configuration

Under the System Configuration users are given the flexibility of enabling their choice of included components on the board as shown in **Figure 4-4**. Each component of the board is listed where users can enable or disable a component according to their design by simply marking a check or removing the check in the field provided. If the component is enabled, the DE1-SoC System Builder will automatically generate the associated pin assignments including the pin name, pin location, pin direction, and I/O standard.

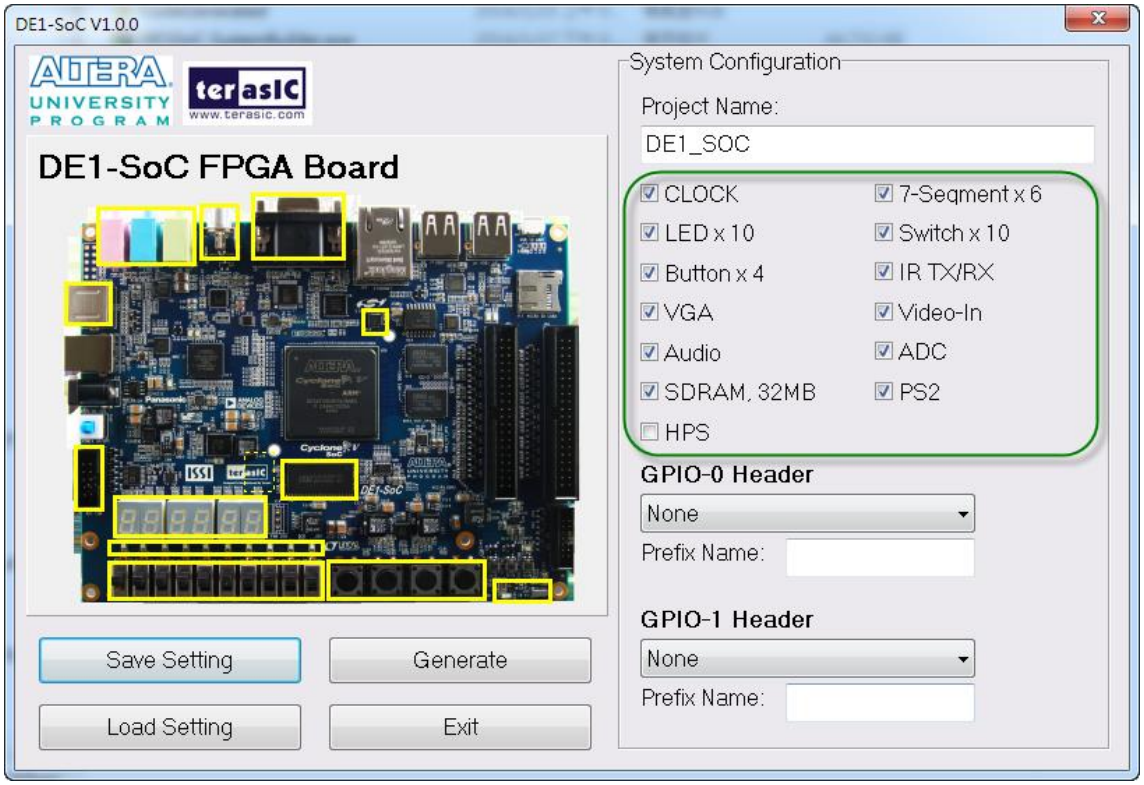


Figure 4-4 System Configuration Group

■ GPIO Expansion

Users can connect GPIO daughter cards onto the GPIO connector located on the development board shown in **Figure 4-5**. Select the daughter card you wish to add to your design under the appropriate GPIO connector to which the daughter card is connected. The System Builder will automatically generate the associated pin assignment including pin name, pin location, pin direction, and I/O standard.

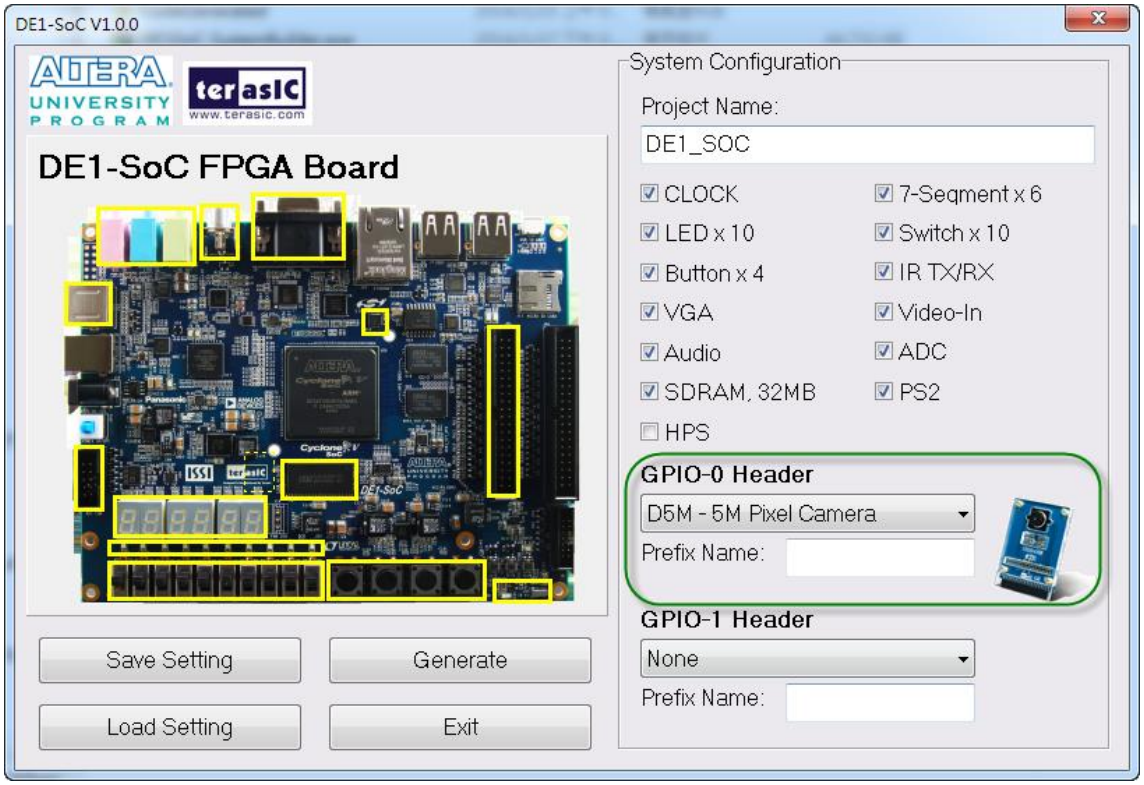


Figure 4-5 GPIO Expansion Group

The "Prefix Name" is an optional feature that denotes the pin name of the daughter card assigned in your design. Users may leave this field empty.

■ Project Setting Management

The DE1-SoC System Builder also provides functions to restore default setting, load a setting, and save users' board configuration file shown in **Figure 4-6**. Users can save the current board configuration information into a .cfg file and load it to the DE1-SoC System Builder.

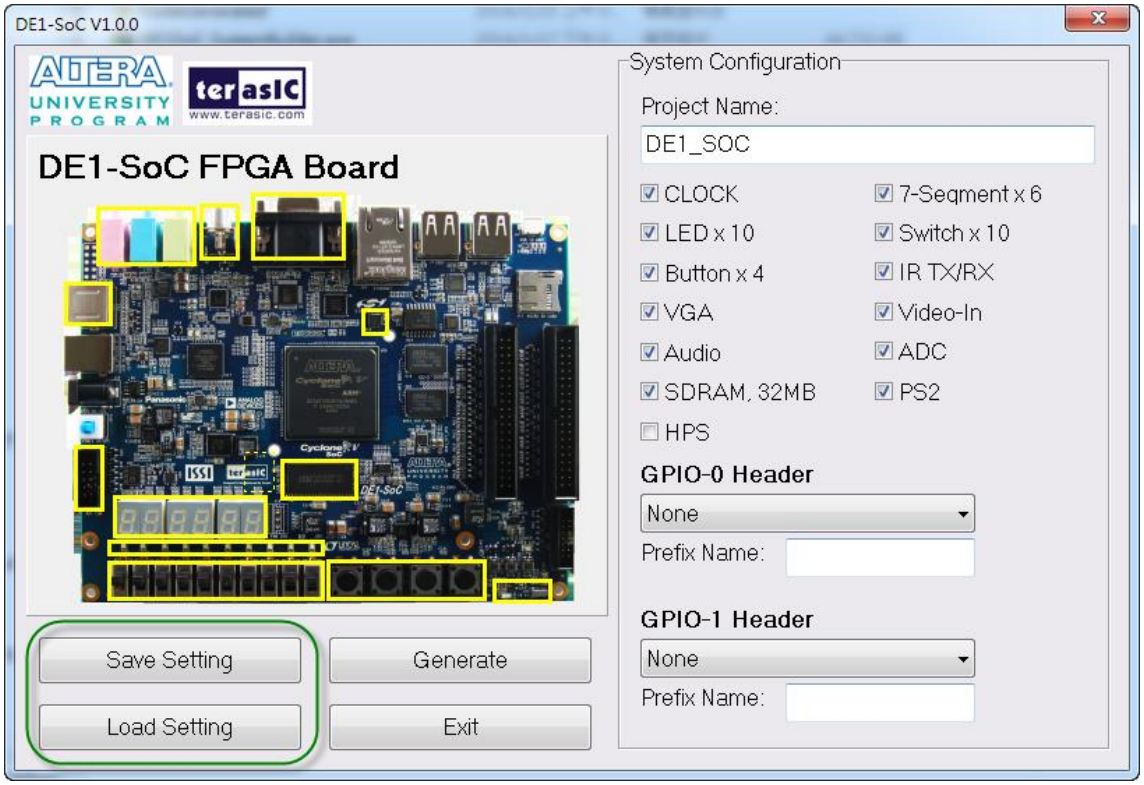


Figure 4-6 Project Settings

■ Project Generation

When users press the *Generate* button, the DE1-SoC System Builder will generate the corresponding Quartus II files and documents as listed in the Table 4-1:

Table 4-1 The files generated by DE1-SoC System Builder

No.	*Filename*	*Description*
1	<Project name>.v	Top level Verilog HDL file for Quartus II
2	<Project name>.qpf	Quartus II Project File
3	<Project name>.qsf	Quartus II Setting File
4	<Project name>.sdc	Synopsis Design Constraints file for Quartus II
5	<Project name>.htm	Pin Assignment Document

Users can use Quartus II software to add custom logic into the project and compile the project to generate the SRAM Object File (.sof).

Chapter 5

Examples For FPGA

This chapter provides a number of examples of advanced circuits implemented by RTL or Qsys on the DE1-SoC board. These circuits provide demonstrations of the major features which connected to FPGA interface on the board, such as audio, SDRAM and IR receiver. All of the associated files can be found in the Demonstrations/FPGA folder on the DE1-SoC System CD.

■ Installing the Demonstrations

To install the demonstrations on your computer:

Copy the directory Demonstrations into a local directory of your choice. It is important to ensure that the path to your local directory contains no spaces – otherwise, the Nios II software will not work. Note Quartus II v13 or later is required for all DE1-SoC demonstrations to support Cyclone V SoC device.

5.1 DE1-SoC Factory Configuration

The DE1-SoC board is shipped from the factory with a default configuration bit-stream that demonstrates some of the basic features of the board. The setup required for this demonstration, and the locations of its files are shown below.

■ Demonstration Setup, File Locations, and Instructions

- Project directory: DE1_SoC_Default
- Bit stream used: DE1_SoC_Default.sof or DE1_SoC_Default.jic
- Power on the DE1-SoC board, with the USB cable connected to the USB Blaster port. If necessary (that is, if the default factory configuration of the DE1-SoC board is not currently stored in EPCQ device), download the bit stream to the board by using JTAG programming
- You should now be able to observe that the 7-segment displays are displaying a sequence of characters, and the red LEDs are flashing.
- Optionally connect a VGA display to the VGA D-SUB connector. When connected, the VGA

display should show a color picture

- Optionally connect a powered speaker to the stereo audio-out jack. Press KEY[1] to hear a 1 kHz humming sound from the audio-out port.
- There is a demo_batch folder in the project. It is able to load the bit stream into the FPGA , programming or erasing .jic file to EPCQ by executing test.bat file as shown in the **Figure 5-1**.If user want to download the new design into the EPCQ, the easy method is to copy the new .sof file into the demo_batch folder and Execute the test.bat. Select the option “2” to covert the .sof to .jic firstly. Then using the option”3” to program .jic file into EPCQ.

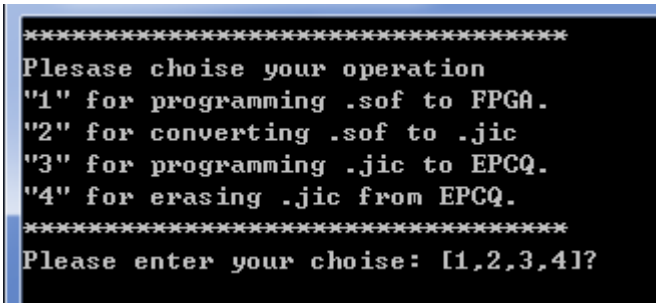


Figure 5-1 Batch file for download FPGA and EPCQ

5.2 Audio Recording and Playing

This demonstration shows how to implement an audio recorder and player using the DE1-SoC board with the built-in Audio CODEC chip. This demonstration is developed based on Qsys and Eclipse. **Figure 5-2** shows the man-machine interface of this demonstration. Two push-buttons and four slide switches are used for users to configure this audio system: SW0 is used to specify recording source to be Line-in or MIC-In. SW1, SW2, and SW3 are used to specify recording sample rate as 96K, 48K, 44.1K, 32K, or 8K. **Table 5-1** and **Table 5-2** summarize the usage of Slide switches for configuring the audio recorder and player.

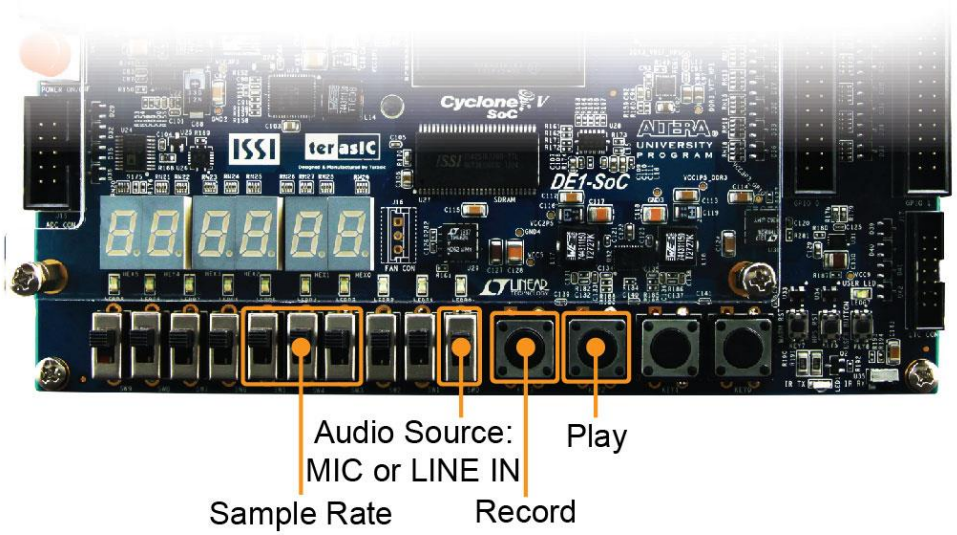


Figure 5-2 Man-Machine Interface of Audio Recorder and Player

Figure 5-3 shows the block diagram of the Audio Recorder and Player design. There are hardware and software parts in the block diagram. The software part stores the Nios II program in the on-chip memory. The software part is built by Eclipse in C programming language. The hardware part is built by Qsys under Quartus II. The hardware part includes all the other blocks. The "AUDIO Controller" is a user-defined Qsys component. It is designed to send audio data to the audio chip or receive audio data from the audio chip.

The audio chip is programmed through I2C protocol which is implemented in C code. The I2C pins from audio chip are connected to Qsys System Interconnect Fabric through PIO controllers. In this example, the audio chip is configured in Master Mode. The audio interface is configured as I2S and 16-bit mode. 18.432MHz clock generated by the PLL is connected to the MCLK/XTI pin of the audio chip through the AUDIO Controller.

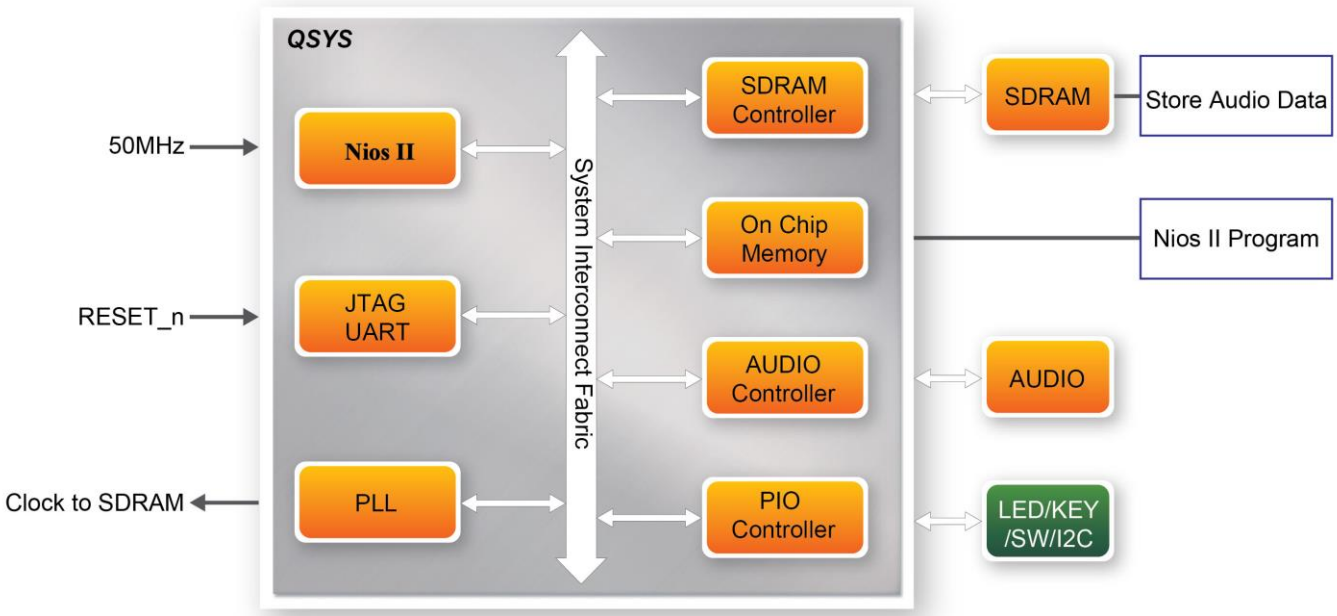


Figure 5-3 Block diagram of the audio recorder and player

■ Demonstration Setup, File Locations, and Instructions

- Hardware Project directory: DE1_SoC _Audio
- Bit stream used: DE1_SoC _Audio.sof
- Software Project directory: DE1_SoC _Audio\software
- Connect an Audio Source to the LINE-IN port of the DE1-SoC board.
- Connect a Microphone to MIC-IN port on the DE1-SoC board.
- Connect a speaker or headset to LINE-OUT port on the DE1-SoC board.
- Load the bit stream into FPGA. (note *1)
- Load the Software Execution File into FPGA. (note *1)
- Configure audio with the Slide switches SW0 as shown in **Table 5-1**.
- Press KEY3 on the DE1-SoC board to start/stop audio recording (note *2)
- Press KEY2 on the DE1-SoC board to start/stop audio playing (note *3)

Table 5-1 Slide switches usage for audio source

Slide Switches	*0 – DOWN Position*	*1 – UP Position*
SW0	**Audio is from MIC**	**Audio is from LINE-IN**

Table 5-2 Slide switch setting for sample rate switching for audio recorder and player

SW5 (0 – DOWN; 1- UP)	*SW4 (0 – DOWN; 1-UP)*	*SW3 (0 – DOWN; 1-UP)*	*Sample Rate*
0	0	0	96K
0	0	1	48K
0	1	0	44.1K
0	1	1	32K
1	0	0	8K
Unlisted combination			96K

Note:

(1). Execute DE1_SoC _Audio \demo_batch\ DE1-SoC _Audio.bat will download .sof and .elf files.

(2). Recording process will stop if audio buffer is full.

(3). Playing process will stop if audio data is played completely.

5.3 A Karaoke Machine

This demonstration uses the microphone-in, line-in, and line-out ports on the DE1-SOC board to create a Karaoke Machine application. The WM8731 CODEC is configured in the master mode, with which the audio CODEC generates AD/DA serial bit clock (BCK) and the left/right channel clock (LRCK) automatically. As indicated in **Figure 5-4**, the I2C interface is used to configure the Audio CODEC. The sample rate and gain of the CODEC are set in this manner, and the data input from the line-in port is then mixed with the microphone-in port and the result is sent to the line-out port.

For this demonstration the sample rate is set to 48kHz. Pressing the pushbutton KEY0 reconfigures the gain of the audio CODEC via I2C bus, cycling within ten predefined gain values (volume levels) provided by the device.

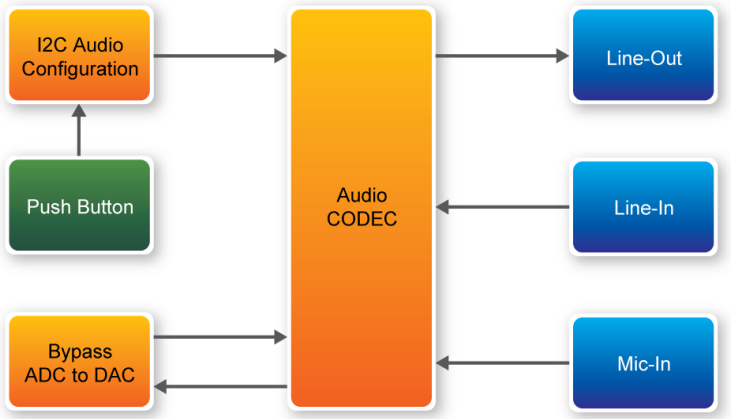


Figure 5-4 Block diagram of the Karaoke Machine demonstration

■ Demonstration Setup, File Locations, and Instructions

- Project directory: DE1_SOC_i2sound
- Bit stream used: DE1_SOC_i2sound.sof
- Connect a microphone to the microphone-in port (pink color) on the DE1-SOC board
- Connect the audio output of a music-player, such as an MP3 player or computer, to the line-in port (blue color) on the DE1-SOC board
- Connect a headset/speaker to the line-out port (green color) on the DE1-SOC board
- Load the bit stream into the FPGA by execute the batch file 'DE1_SOC_i2sound' under the DE1_SOC_i2sound\demo_batch folder
- You should be able to hear a mixture of the microphone sound and the sound from the music player
- Press KEY0 to adjust the volume; it cycles between volume levels 0 to 9

Figure 5-5 illustrates the setup for this demonstration.

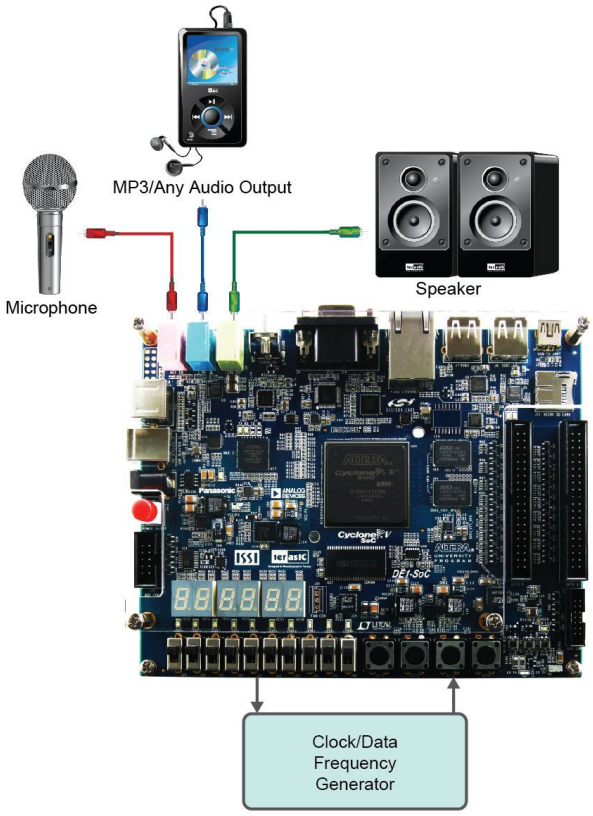


Figure 5-5 Setup for the Karaoke Machine

5.4 SDRAM Test by Nios II

Many applications use SDRAM to provide temporary storage. In this demonstration hardware and software designs are provided to illustrate how to perform memory access in QSYS. We describe how the Altera's SDRAM Controller IP is used to access a SDRAM, and how the Nios II processor is used to read and write the SDRAM for hardware verification. The SDRAM controller handles the complex aspects of using SDRAM by initializing the memory devices, managing SDRAM banks, and keeping the devices refreshed at appropriate intervals.

■ System Block Diagram

Figure 5-6 shows the system block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The SDRAM controller is configured as a 64MB controller. The working frequency of the SDRAM controller is 100MHz, and the Nios II program is running in the on-chip memory.

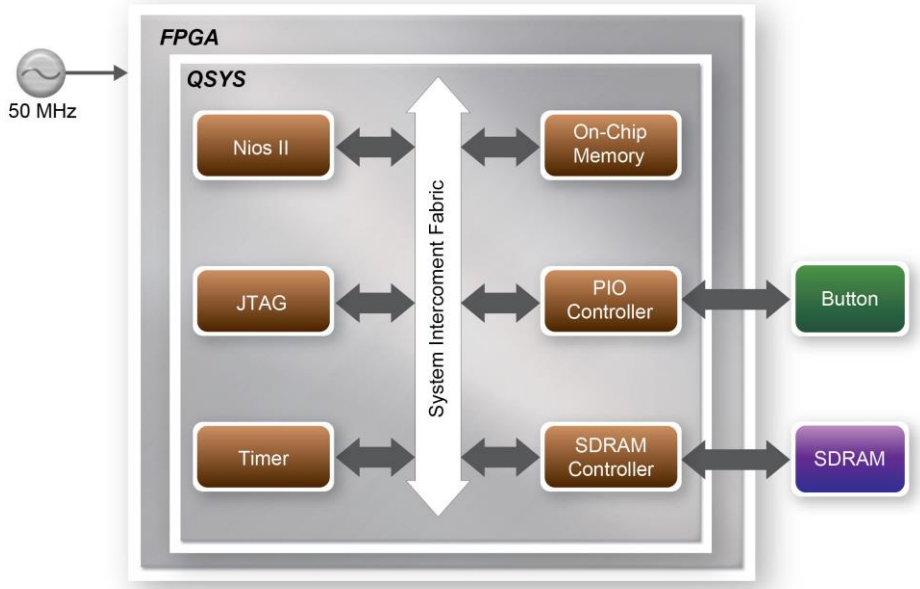


Figure 5-6 Block diagram of the SDRAM Basic Demonstration

The system flow is controlled by a Nios II program. First, the Nios II program writes test patterns into the whole 64MB of SDRAM. Then, it calls Nios II system function, alt_dcache_flush_all, to make sure all data has been written to SDRAM. Finally, it reads data from SDRAM for data verification. The program will show progress in JTAG-Terminal when writing/reading data to/from the SDRAM. When verification process is completed, the result is displayed in the JTAG-Terminal.

■ Design Tools

- Quartus II 13.1
- Nios II Eclipse 13.1

■ Demonstration Source Code

- Quartus Project directory: DE1_SoC_SDRAM_Nios_Test
- Nios II Eclipse: DE1_SoC_SDRAM_Nios_Test \Software

■ Nios II Project Compilation

- Before you attempt to compile the reference design under Nios II Eclipse, make sure the project is cleaned first by clicking ‘Clean’ from the ‘Project’ menu of Nios II Eclipse.

■ Demonstration Batch File

Demo Batch File Folder:

DE1_SoC_SDRAM_Nios_Test \demo_batch

The demo batch file includes following files:

- Batch File for USB-Blaster (II) : DE1_SoC_SDRAM_Nios_Test.bat, DE1_SoC_SDRAM_Nios_Test_bashrc
- FPGA Configure File : DE1_SoC_SDRAM_Nios_Test.sof
- Nios II Program: DE1_SoC_SDRAM_Nios_Test.elf

■ Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Power on the DE1-SoC board.
- Use USB cable to connect PC and the DE1-SoC board (J13) and install USB Blaster driver if necessary.
  Execute the demo batch file "DE1_SoC_SDRAM_Nios_Test.bat" for USB-Blaster II under the batch file folder, DE1_SoC_SDRAM_Nios_Test\demo_batch
- After Nios II program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.
- Press **KEY3~KEY0** of the DE1-SoC board to start SDRAM verify process. Press **KEY0** for continued test**.**
- The program will display progressing and result information, as shown in **Figure 5-7**.

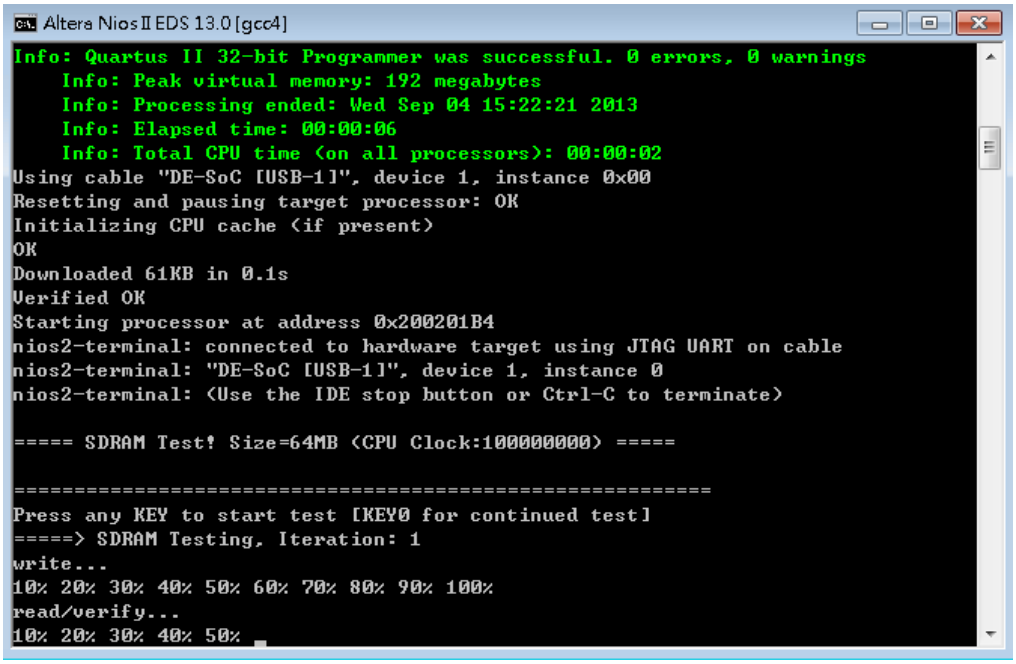


Figure 5-7 Display Progress and Result Information for the SDRAM Demonstration

5.5 SDRAM RTL Test

This demonstration presents a memory test function on the bank of SDRAM on the DE1-SoC board. The memory size of the SDRAM bank is 64MB and all the test codes on this demonstration are written in Verilog HDL.

- **Function Block Diagram**

Figure 5-8 shows the function block diagram of this demonstration. The controller uses 50 MHz as a reference clock, generates one 100 MHz clock as memory clock.

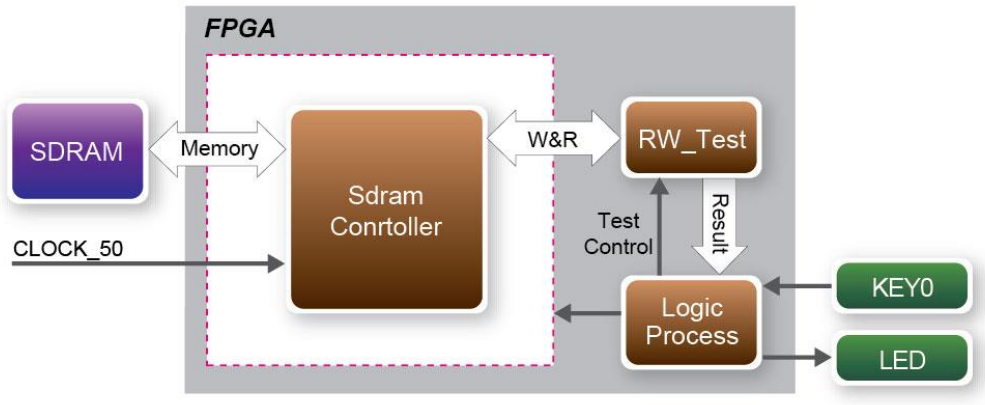


Figure 5-8 Block Diagram of the SDRAM Demonstration

RW_test modules read and write the entire memory space of the SDRAM through the interface of the controller. In this project, the read/write test module will first write the entire memory and then compare the read back data with the regenerated data (the same sequence as the write data). KEY0 will trigger test control signals for the SDRAM, and the LEDs will indicate the test results according to **Table 5-3.**

■ Design Tools

- Quartus 13.1

■ Demonstration Source Code

- Project directory: DE1_SoC_SDRAM_RTL_Test
- Bit stream used: DE1_SoC_SDRAM_RTL_Test.sof

■ Demonstration Batch File

Demo Batch File Folder: DE1_SoC_SDRAM_RTL_Test\demo_batch

The demo batch file includes following files:

- Batch File: DE1_SoC_SDRAM_RTL_Test.bat
- FPGA Configure File: DE1_SoC_SDRAM_RTL_Test.sof

■ Demonstration Setup

- Make sure Quartus II is installed on your PC.
- Connect the USB cable to the USB Blaster connector on the DE1_SoC board and host PC.

- Power on the DE1_SoC board.
- Execute the demo batch file " DE1_SoC_SDRAM_RTL_Test.bat" under the batch file folder, DE1_SoC_SDRAM_RTL_Test \demo_batch.
- Press **KEY0** on the DE1_SoC board to start the verification process. When **KEY0** is pressed, the **LEDs** (**LEDR** [2:0]) should turn on. At the instant of releasing **KEY0**, **LEDR1, LEDR2** should start blinking. After approximately 8 seconds, **LEDR1** should stop blinking and stay on to indicate that the SDRAM has passed the test, respectively. Table 5-3 lists the **LED** indicators.
- If **LEDR2** is not blinking, it means 50MHz clock source is not working.
- If **LEDR1** fail to remain on after 8 seconds, the corresponding SDRAM test has failed.
- Press **KEY0** again to regenerate the test control signals for a repeat test.

Table 5-3 LED Indicators

Table 5-4NAME	Description
LEDR0	Reset
LEDR1	If light, SDRAM test pass
LEDR2	Blinks

5.6 TV Box Demonstration

This demonstration plays video and audio input from a DVD player using the VGA output, audio CODEC, and one TV decoder on the DE1-SoC board. Figure 5-9 shows the block diagram of the design. There are two major blocks in the circuit, called I2C_AV_Config and TV_to_VGA. The TV_to_VGA block consists of the ITU-R 656 Decoder, SDRAM Frame Buffer, YUV422 to YUV444, YCbCr to RGB, and VGA Controller. The figure also shows the TV Decoder (ADV7180) and the VGA DAC (ADV7123) chips used.

As soon as the bit stream is downloaded into the FPGA, the register values of the TV Decoder chip are used to configure the TV decoder via the I2C_AV_Config block, which uses the I2C protocol to communicate with the TV Decoder chip. Following the power-on sequence, the TV Decoder chip will be unstable for a time period; the Lock Detector is responsible for detecting this instability.

The ITU-R 656 Decoder block extracts YcrCb 4:2:2 (YUV 4:2:2) video signals from the ITU-R 656 data stream sent from the TV Decoder. It also generates a data valid control signal indicating the valid period of data output. Because the video signal from the TV Decoder is interlaced, we need to perform de-interlacing on the data source. We used the SDRAM Frame Buffer and a field selection multiplexer (MUX) which is controlled by the VGA controller to perform the de-interlacing operation. Internally, the VGA Controller generates data request and odd/even selection signals to the SDRAM Frame Buffer and filed selection multiplexer (MUX). The YUV422 to YUV444 block

converts the selected YcrCb 4:2:2 (YUV 4:2:2) video data to the YcrCb 4:4:4 (YUV 4:4:4) video data format.

Finally, the YcrCb_to_RGB block converts the YcrCb data into RGB data output. The VGA Controller block generates standard VGA synchronous signals VGA_HS and VGA_VS to enable the display on a VGA monitor.

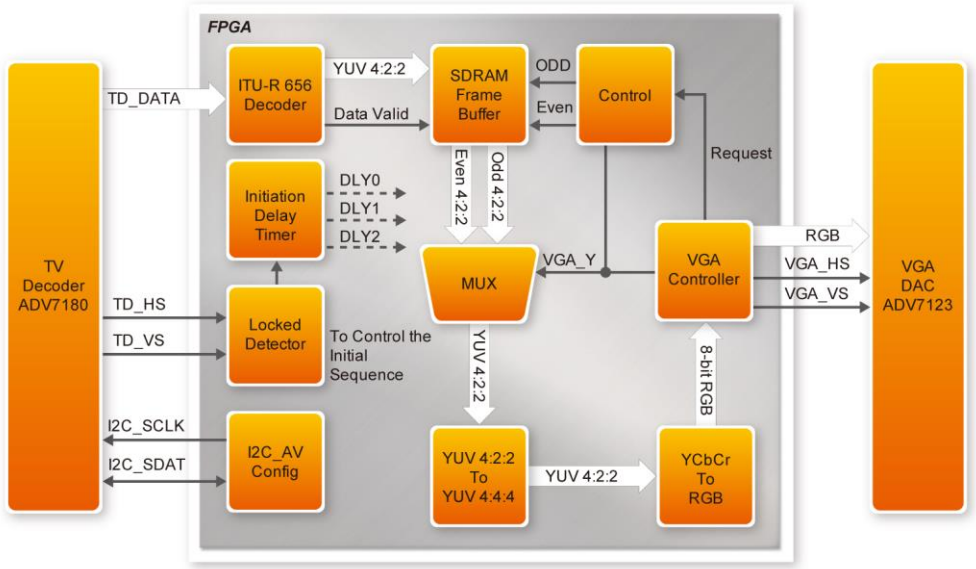


Figure 5-9 Block diagram of the TV box demonstration

Demonstration Source Code

- Project directory: DE1_SoC_TV
- Bit stream used: DE1_SoC_TV.sof

Demonstration Batch File

Demo Batch File Folder: DE1_SoC_TV \demo_batch

The demo batch file includes the following files:

- Batch File: DE1_SoC_TV.bat
- FPGA Configure File : DE1_SoC_TV.sof

Demonstration Setup, File Locations, and Instructions

- Connect a DVD player's composite video output (yellow plug) to the Video-In RCA jack (J6) of

the DE1-SoC board (See **Figure 5-10**). The DVD player has to be configured to provide:

- NTSC output
- 60Hz refresh rate
- 4:3 aspect ratio
- Non-progressive video
- Connect the VGA output of the DE1-SoC board to a VGA monitor (both LCD and CRT type of monitors should work).
- Connect the audio output of the DVD player to the line-in port of the DE1-SoC board and connect a speaker to the line-out port. If the audio output jacks from the DVD player are RCA type, then an adaptor will be needed to convert to the mini-stereo plug supported on the DE1-SoC board; this is the same type of plug supported on most computers.
- Load the bit stream into FPGA by execute the batch file 'DE1_SoC_TV.bat' under DE1_SoC_TV \demo_batch\ folder. Press KEY0 on the DE1-SoC board to reset the circuit

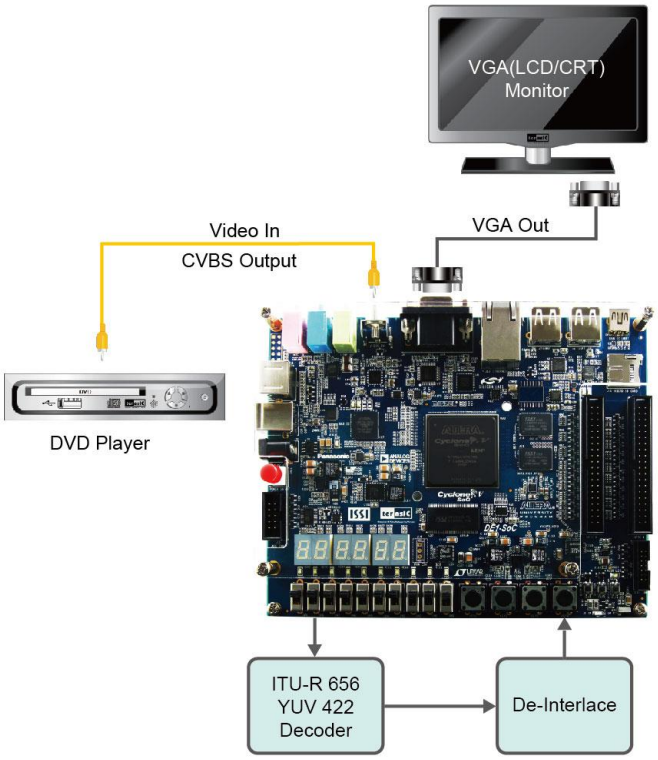


Figure 5-10 Setup for the TV box demonstration

5.7 PS/2 Mouse Demonstration

We offer this simple PS/2 controller coded in Verilog HDL to demonstrate bidirectional

communication between PS/2 controller and the device, the PS/2 mouse. You can treat it as a how-to basis and develop your own controller that could accomplish more sophisticated instructions, like setting the sampling rate or resolution, which needs to transfer two data bytes.

For detailed information about the PS/2 protocol, please perform an appropriate search on various educational web sites. Here we give a brief introduction:

■ Outline

PS/2 protocol uses two wires for bidirectional communication, one clock line and one data line. The PS/2 controller always has total control over the transmission line, but the PS/2 device generates clock signal during data transmission.

■ Data transmit from the device to controller

After sending an enabling instruction to the PS/2 mouse at stream mode, the device starts to send displacement data out, which consists of 33 bits. The frame data is cut into three similar slices, each of them containing a start bit (always zero) and eight data bits (with LSB first), one parity check bit (odd check), and one stop bit (always one).

PS/2 controller samples the data line at the falling edge of the PS/2 clock signal. This could easily be implemented using a shift register of 33 bits, but be cautious with the clock domain crossing problem.

■ Data transmit from the controller to device

Whenever the controller wants to transmit data to device, it first pulls the clock line low for more than one clock cycle to inhibit the current transmit process or to indicate the start of a new transmit process, which usually be called as inhibit state. After that, it pulls low the data line then release the clock line, and this is called the request state. The rising edge on the clock line formed by the release action can also be used to indicate the sample time point as for a 'start bit. The device will detect this succession and generates a clock sequence in less than 10ms time. The transmit data consists of 12bits, one start bit (as explained before), eight data bits, one parity check bit (odd check), one stop bit (always one), and one acknowledge bit (always zero). After sending out the parity check bit, the controller should release the data line, and the device will detect any state change on the data line in the next clock cycle. If there’s no change on the data line for one clock cycle, the device will pull low the data line again as an acknowledgement which means that the data is correctly received.

After the power on cycle of the PS/2 mouse, it enters into stream mode automatically and disable data transmit unless an enabling instruction is received. **Figure 5-11** shows the waveform while communication happening on two lines.

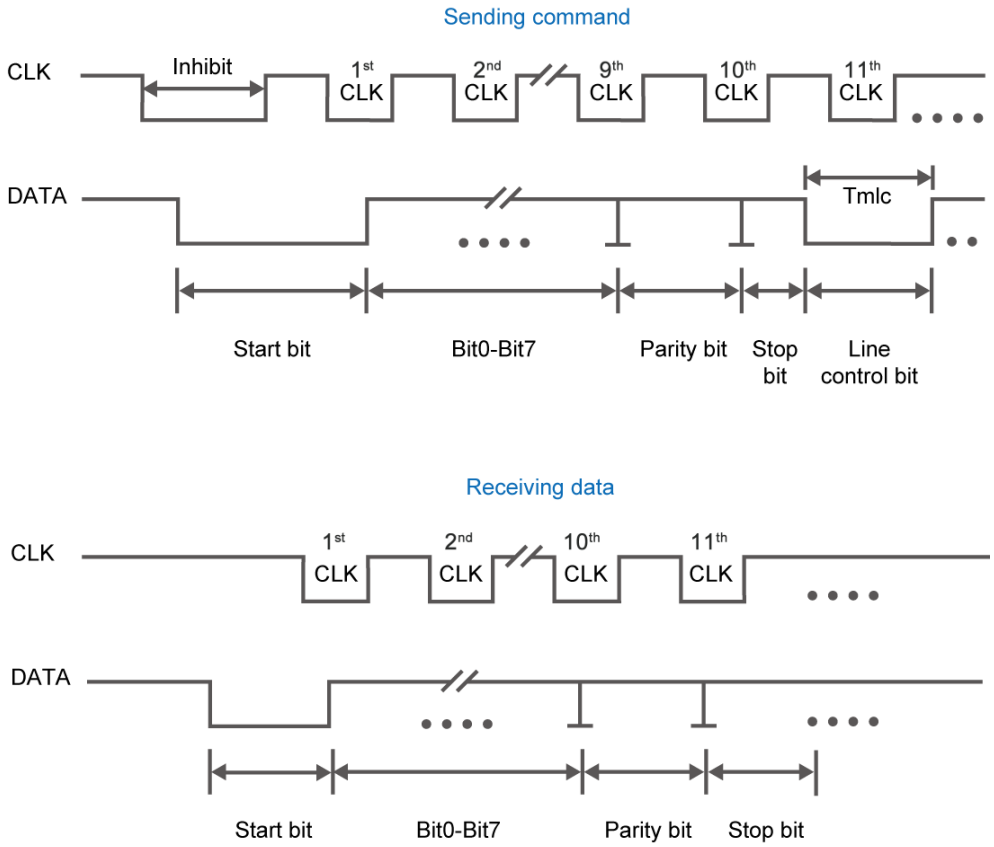


Figure 5-11 Waveforms on two lines while communication taking place

Demonstration Source Code

- Project directory: DE1_SoC_PS2_DEMO
- Bit stream used: DE1_SoC_PS2_DEMO.sof

Demonstration Batch File

Demo Batch File Folder: DE1_SoC_PS2_DEMO \demo_batch

The demo batch file includes the following files:

- Batch File: DE1_SoC_PS2_DEMO.bat
- FPGA Configure File : DE1_SoC_PS2_DEMO.sof

Demonstration Setup, File Locations, and Instructions

- Load the bit stream into FPGA by executing DE1_SoC_PS2_DEMO \demo_batch\ DE1_SoC_PS2_DEMO.bat
- Plug in the PS/2 mouse
- Press KEY[0] for enabling data transfer
- Press KEY[1] to clear the display data cache
- You should see digital changes on 7-segment display when the PS/2 mouse moves, and the LEDR[2:0] will blink respectively when the left-button, right-button or middle-button is pressed. Table 5-5 gives the detailed information.

Table 5-5 Detailed information of the indicators

Indicator Name	*Description*
LEDR[0]	**Left button press indicator**
LEDR[1]	**Right button press indicator**
LEDR[2]	**Middle button press indicator**
HEX0	**Low byte of X displacement**
HEX1	**High byte of X displacement**
HEX2	**Low byte of Y displacement**
HEX3	**High byte of Y displacement**

5.8 IR Emitter LED and Receiver Demonstration

In this demonstration we show a simple example of using IR Emitter LED and IR receiver. All the codes in this demonstration are coding by verilog HDL.

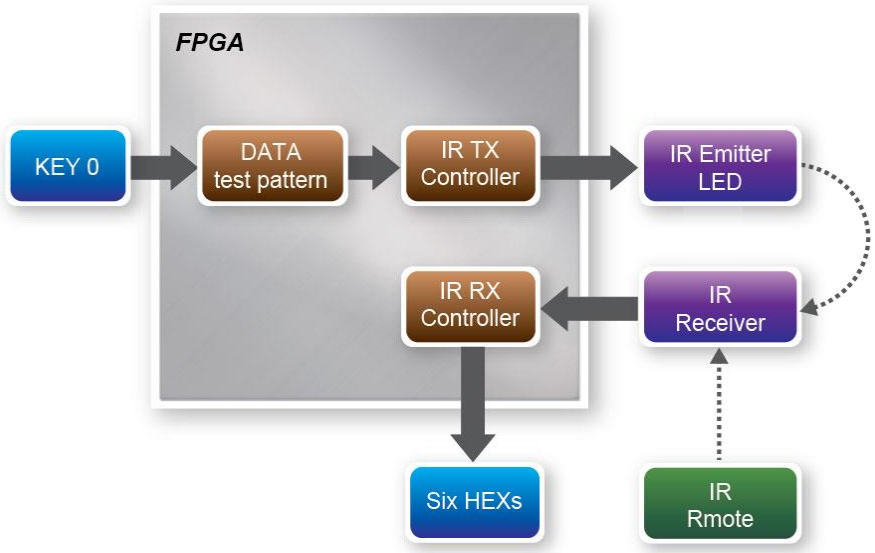


Figure 5-12 Block Diagram of the IR Emitter LED and Receiver Demonstration

Figure 5-12 shows the block diagram of the design. It mainly implement a IR TX Controller and a IR RX Controller. When KEY0 is pressed, Data test pattern generator continuously generates data to the IR TX Controller. When IR TX Controller works, it will format the sending data into NEC IR transmission protocol and send it out through IR emitter LED. IR receiver will decode the received data and display it on six HEXs. Also, user can use a remote controller to sending data to IR Receiver. The main function of IR TX /RX controller and IR remote in this demonstration will be described in below:

■ IR TX Controller

User can input 8-bit address and 8-bits command into IR TX Controller. IR TX Controller will encode the address and command first , and send it out according to NEC IR transmission protocol through IR emitter LED. Note that the input clock of the Controller should be 50MHz.

The NEC IR transmission protocol uses pulse distance encoding of the message bits. Each pulse burst is 562.5µs in length, at a carrier frequency of 38kHz (26.3µs). As shown in Figure 5-13,Logical bits are transmitted as follows:

- Logical '0' – a 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms

- Logical '1' – a 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms

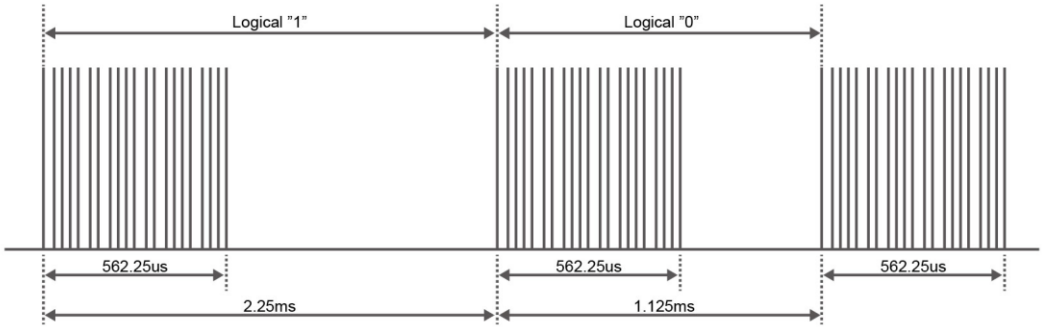


Figure 5-13 Logical "1"and Logical "0"

Figure 5-14 shows the frame of the protocol. Protocol will send a lead code first, a 9ms leading pulse burst followed by a 4.5msThe second inversed data is sent to verify the accuracy of the information received. At last, a final 562.5µs pulse burst to signify the end of message transmission.. Because every time it is sent inversed data, the overall transmission time is constant.

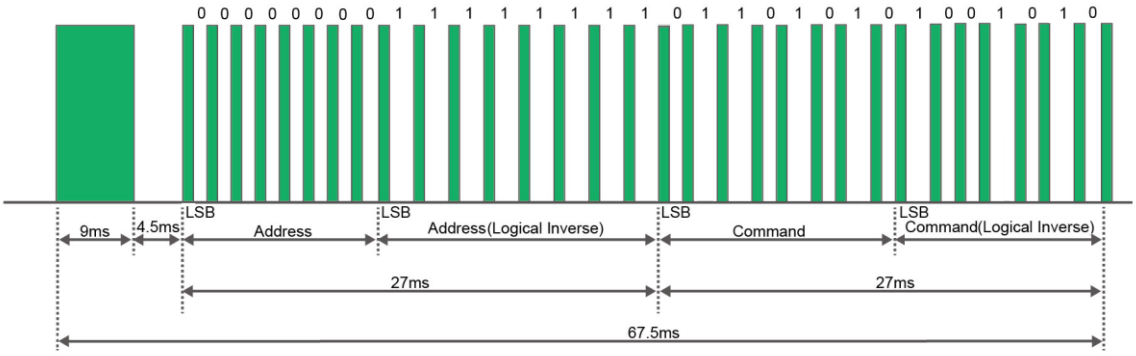


Figure 5-14 Typical frame of NEC protocol

Note: IR Receiver receives the signal a inverted value(e.g. IR TX Controller send a lead code 9 ms high then 4.5 ms low, IR Receiver will receive a 9 ms low then 4.5 ms high lead code).

■ IR Remote

When a key on the remote controller(See Figure 5-15) is pressed, the remote controller will emit a

standard frame, shown in Table 5-6 The beginning of the frame is the lead code represents the start bit, and then is the key-related information, and the last 1 bit end code represents the end of the frame.(This frame is descript the signal which IR Receiver Received)

Figure 5-15 Remote controller

Table 5-6 Key code information for each Key on remote controller

Key	*Key Code*	*Key*	*Key Code*	*Key*	*Key Code*	*Key*	*Key Code*
A	0x0F	B	0x13	C	0x10	⏻	0x12
1	0x01	2	0x02	3	0x03	▲	0x1A
4	0x04	5	0x05	6	0x06	▼	0x1E
7	0x07	8	0x08	9	0x09	▲	0x1B
⏹	0x11	0	0x00	↩	0x17	▼	0x1F
⏯	0x16	◀	0x14	▶	0x18	🔇	0x0C

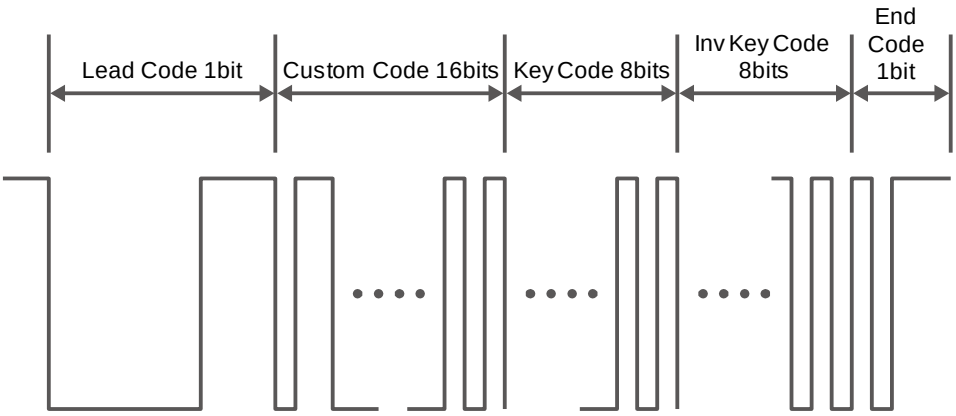


Figure 5-16 The transmitting frame of the IR remote controller

■ IR RX Controller

In this demo, the IP of IR receiver controller is implemented in the FPGA. As **Figure 5-17** show, it includes Code Detector, State Machine, and Shift Register. First, the IR receiver demodulates the signal inputs to Code Detector block .The Code Detector block will check the Lead Code and feedback the examination result to State Machine block.

The State Machine block will change the state from IDLE to GUIDANCE once the Lead code is detected. Once the Code Detector has detected the Custom Code status, the current state will change from GUIDANCE to DATAREAD state. At this state, the Code Detector will save the receiving data and output to Shift Register then displays it on 7-segment displays. **Figure 5-18** shows the state shift diagram of State Machine block. Note that the input clock should be 50MHz.

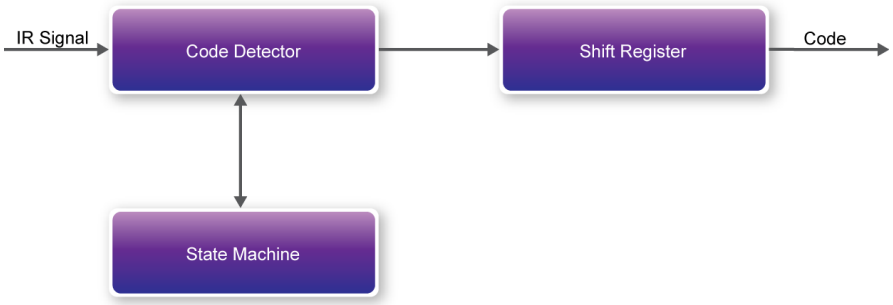


Figure 5-17 The IR Receiver controller

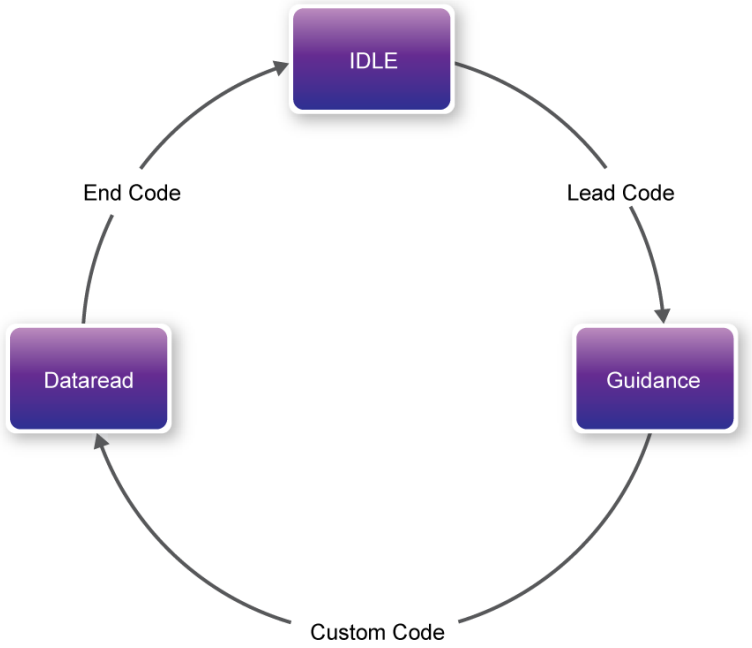


Figure 5-18 State shift diagram of State Machine

Demonstration Source Code

- Project directory: DE1_SoC_IR
- Bit stream used: DE1_SOC_IR.sof

Demonstration Batch File

Demo Batch File Folder: DE1_SoC_IR \demo_batch

The demo batch file includes the following files:

- Batch File: DE1_SoC_IR.bat
- FPGA Configure File : DE1_SOC_IR.sof

Demonstration Setup, File Locations, and Instructions

- Load the bit stream into FPGA by executing DE1_SoC_IR \demo_batch\ DE1_SoC_IR.bat
- Press KEY[0] to enable the continuously pattern sending out by IR TX Controller.
- Observe the six HEXs (See **Table 5-7** for detail)
- Releasing KEY[0] to stop the IR TX.
- Point the IR receiver with the remote-controller and press any button
- Observe the six HEXs(See **Table 5-7** for detail)
- And User can using Multi DE1_SoC boards to do more IR test between boards.

Table 5-7 Detailed information of the indicators

Indicator Name	*Description*
HEX5	Inversed high byte of DATA(Key Code)
HEX4	Inversed low byte of DATA(Key Code)
HEX3	High byte of ADDRESS(Custom Code)
HEX2	Low byte of ADDRESS(Custom Code)
HEX1	High byte of DATA(Key Code)
HEX0	Low byte of DATA (Key Code)

5.9 ADC Reading

This demonstration illustrates steps which can be used to evaluate the performance of the 8-channel 12-bit A/D Converter ADC7928. The DC 5.0V on the 2x5 header is used to drive the analog signals and by using a trimmer potentiometer, the voltage can be adjusted within the range of 0~5.0V. The 12-bit voltage measurements are indicated on the NIOS II console. **Figure 5-19** shows the block diagram of this demonstration.

Note that，the input voltage range is 0-5V，and if your input voltage is -2.5-2.5V，you can make the pre-scale circuit to adjust their range to 0-5V.

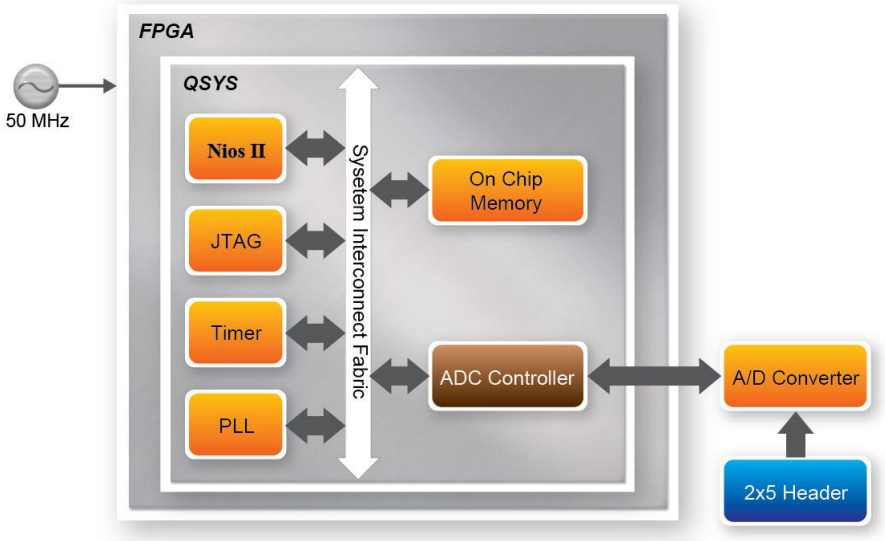


Figure 5-19 ADC Reading Block Diagram

Figure 5-20 depicts the pin arrangement of the 2x5 header. In this demonstration, this header is the input source of ADC convertor. Users can connect a trimmer to the specified ADC channel (ADC_IN0 ~ ADC_IN7) that provides voltage to the ADC convert. Then FPGA will read the associated register in the convertor via serial interface and translates it to voltage value displayed on the NIOS II console

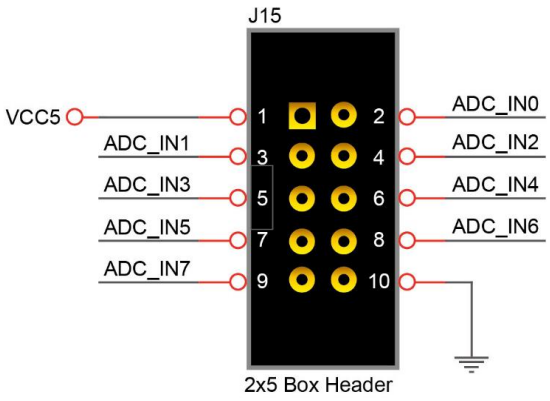


Figure 5-20 ADC Pin distribution of the 2x5 Header

■ System Requirements

The following items are required for the ADC Reading demonstration

- DE1-SoC board x1
- Trimmer Potentiometer x1
- Wire Strip x3

■ Demonstration File Locations

- Hardware Project directory: DE1_SoC_ADC
- Bit stream used: DE1_SoC_ADC.sof
- Software Project directory: DE1_SoC_ADC software
- Demo batch file : DE1_SoC_ADC\demo_batch\ DE1_SoC_ADC.bat

■ Demonstration Setup and Instructions

- Connect the trimmer to corresponding ADC channel on the 2x5 header as shown in Figure 5-21 to read from, as well as the +5V and GND signals. (Note: the setup shown above is connected ADC channel 0).

- Execute the demo batch file DE1_SoC_ADC.bat to load bit stream and software execution file in FPGA.
- The NIOS II console will display the voltage of the specified channel voltage result information

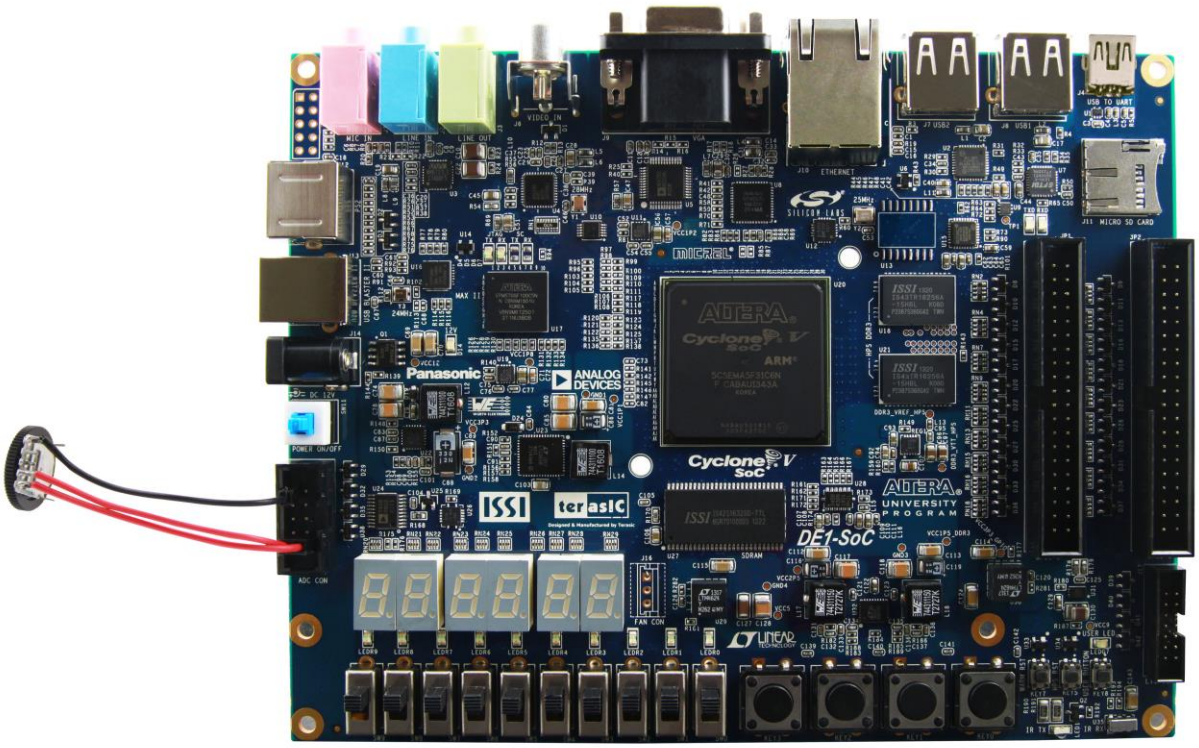


Figure 5-21 ADC Reading hardware setup

Chapter 6

Examples for HPS SoC

This chapter provides a number of C-code examples based on the Altera SoC Linux built by Yocto Project. These examples provide demonstrations of the major features which connected to HPS interface on the board, such as users LED/KEY, I2C interfaced G-sensor and I2C MUX. All of the associated files can be found in the Demonstrations/*SOC* folder in the DE1_SoC System CD. To run Linux on DE1-SOC, Please refer to the chapter 5 "**Running Linux on the DE1-SoC board**" on *DE1-SoC_Getting_Started_Guide.pdf*.

■ Installation of the Demonstrations

To install the demonstrations on your computer:

Copy the directory *Demonstrations* into a local directory of your choice. **Altera SoC EDS v13.1 is required for users to compile the c-code project.**

6.1 Hello Program

This demonstration presents how to develop your first HPS program by using Altera SoC EDS tool. For operation details, please refer to My_First_HPS.pdf in the system CD.

Here are the major procedures to develop and build HPS project.

- Make sure Altera SoC EDS is installed on your PC.
- Create program .c/.h files with a generic text editor
- Create a "Makefile" with a generic text editor
- Build your project under Altera SoC EDS

■ Program File

Here is the main program of this Hello World demo.

```c
#include <stdio.h>

int main(int argc, char **argv) {

  printf("Hello World!\r\n");

  return( 0 );
}
```

- **Makefile**

To compile a project, a Makefile is required. Here is the Makefile used for this demo.

```
#
TARGET = my_first_hps

#
CROSS_COMPILE = arm-linux-gnueabihf-
CFLAGS = -g -Wall  -I $(SOCEDS_DEST_ROOT)/ip/altera/hps/altera_hps/hwlib/include
LDFLAGS =  -g -Wall
CC = $(CROSS_COMPILE)gcc
ARCH= arm

build: $(TARGET)

$(TARGET): main.o
  $(CC) $(LDFLAGS)   $^ -o $@

%.o : %.c
  $(CC) $(CFLAGS) -c $< -o $@

.PHONY: clean
clean:
  rm -f $(TARGET) *.a *.o *~
```

- **Compile**

To compile a project, please launch Altera SoC EDS Command Shell by executing

C:\altera\13.1\embedded\Embedded_Command_Shell.bat

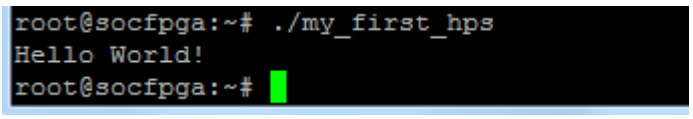
Use the "cd" command to change the current directory to where the Hello World project is located. Then type "make" to build the project. The executable file "**my_first_hps**" will be generated after the compiling process is finished. The "clean all" command can be used to remove all temporary files.

■ Demonstration Source Code

- Build Tool: Altera SoC EDS v13.1
- Project directory: \Demonstration\SoC\my_first_hps
- Binary file: my_first_hps
- Build Command: make ("**make clean**" to remove all temporary files)
- Execute Command: ./my_first_hps
-

■ Demonstration Setup

- Connect USB cable to the USB-to-UART connector (J4) on the DE1_SoC board and host PC.
- Make sure the demo file "**my_first_hps**" is copied into the SD card under the "**/home/root**" folder in Linux.
- Insert the booting micro SD card into the DE1_SoC board.
- Power on the DE1_SoC board.
- Launch PuTTY to connect to the UART port of Putty and type "**root**" to login Altera Yocto Linux.
- In the UART terminal of PuTTY, type "**./my_first_hps**" to start the program, and you will see "Hello World!" message in the terminal.



6.2 Users LED and KEY

This demonstration presents how to control the users LED and KEY by accessing the register of GPIO controller through the memory-mapped device driver. The memory-mapped device driver allows developer to access the system physical memory.

■ Function Block Diagram

Figure 6-1 shows the function block diagram of this demonstration. The users LED and KEY are connected to the **GPIO1** controller in HPS. The behavior of the GPIO controller is controlled by the register in the GPIO controller. The registers can be accessed by application software through the memory-mapped device driver, which is built into Altera SoC Linux.

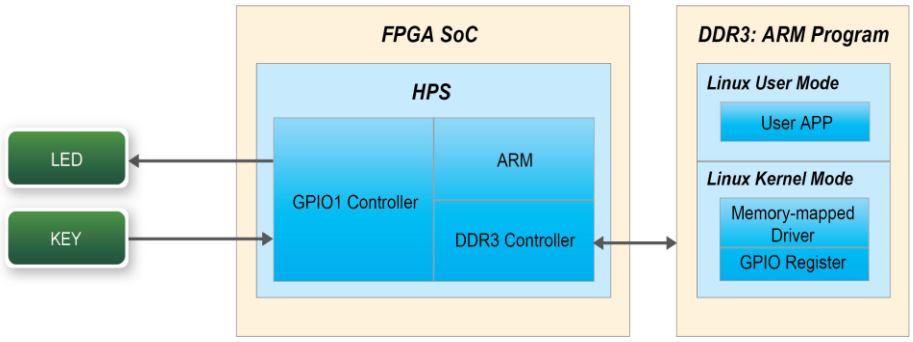


Figure 6-1 Block Diagram of GPIO Demonstration

■ GPIO Interface Block Diagram

The HPS provides three general-purpose I/O (GPIO) interface modules. Figure 6-2 shows the block diagram of the GPIO Interface. GPIO[28..0] is controlled by GPIO0 controller and GPIO[57..29] is controlled by GPIO1 controller. GPIO[70..58] and input-only GPI[13..0] are controlled by GPIO2 controller.

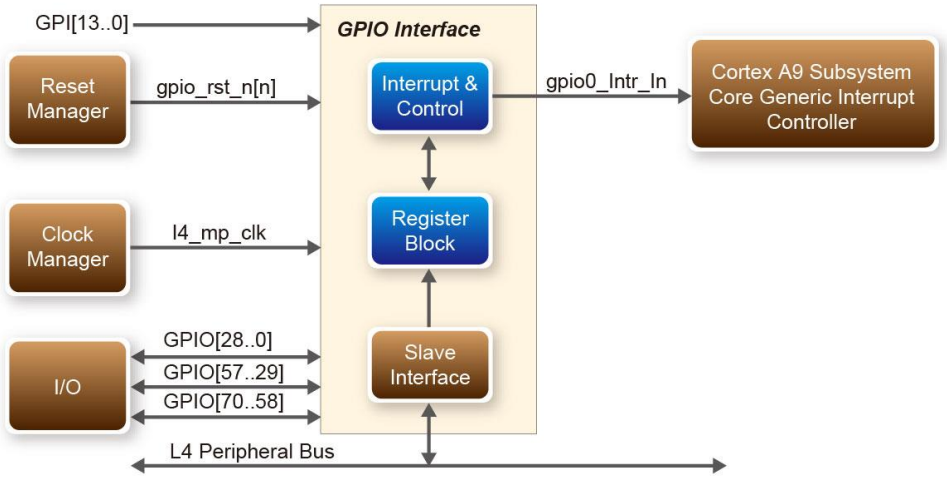


Figure 6-2 Block Diagram of GPIO Interface

■ GPIO Register Block

The behavior of I/O pin is controlled by the registers in the register block. In this demonstration, we only use three 32-bit registers in the GPIO controller. The registers are:

- **gpio_swporta_dr**: used to write output data to output I/O pin
- **gpio_swporta_ddr**: used to configure the direction of I/O pin
- **gpio_ext_porta**: used to read input data of I/O input pin

For LED control, we use **gpio_swporta_ddr** to configure the LED pin as output pin, and drive the pin high or low by writing data to the **gpio_swporta_dr** register. For the **gpio_swporta_ddr** register, the first bit (least significant bit) controls direction of the first IO pin in the associated GPIO controller and the second bit controls the direction of second IO pin in the associated GPIO controller, and so on. The value "1" in the register bit indicates the I/O direction is output, and the value "0" in the register bit indicates the I/O direction is input.

For the **gpio_swporta_dr** register, the first bit controls the output value of first I/O pin in the associated GPIO controller, and the second bit controls the output value of second I/O pin in the associated GPIO controller, and so on. The value "1" in the register bit indicates the output value is high, and the value "0" indicates the output value is low.

The status of KEY can be queried by reading the value of **gpio_ext_porta** register. The first bit represents the input status of first IO pin in the associated GPIO controller, and the second bit represents the input status of second IO pin in the associated GPIO controller, and so on. The value "1" in the register bit indicates the input state is high, and the value "0" indicates the input state is low.

■ GPIO Register Address Mapping

The registers of HPS peripherals are mapped to HPS base address space 0xFC000000 with 64KB size. Registers of GPIO1 controller are mapped to the base address 0xFF208000 with 4KB size, and registers of GPIO2 controller are mapped to the base address 0xFF20A000 with 4KB size, as shown in **Figure 6-3.**

Slave Identifier	Slave Title	Base Address	Size
STM	STM	0xFC000000	48 MB
DAP	DAP	0xFF000000	2 MB
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]NDDATA	NAND controller data	[illegible]	[illegible]
GPIO0	GPIO0	0xFF208000	4 KB
GPIO1	GPIO1	0xFF209000	4 KB
GPIO2	GPIO2	0xFF20A000	4 KB
NAND[illegible]	[illegible] registers	0xFFB80000	64 KB
[illegible]RDATA	FPGA manager [illegible]	[illegible]	[illegible]
CAN0	CAN0 controller registers	0xFFC00000	4 KB
CAN1	CAN1 controller registers	0xFFC01000	4 KB

Figure 6-3 GPIO Address Map

■ Software API

Developers can use the following software API to access the register of GPIO controller.

- open: use to open memory mapped device driver
- mmap: map physical memory to user space
- alt_read_word: read a value from a specified register
- alt_write_word: write a value into a specified register
- munmap: clean up memory mapping
- close: close device driver.

Developers can also use the following MACRO to access the register

- alt_setbits_word: set specified bit value to zero for a specified register
- alt_clrbits_word: set specified bit value to one for a specified register

To use the above API to access register of GPIO controller, the program must include the following header files.

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include "hwlib.h"
#include "socal/socal.h"
#include "socal/hps.h"
```

```
#include "socal/alt_gpio.h"
```

■ LED and KEY Control

Figure 6-4 shows the HPS users LED and KEY pin assignment for the DE1_SoC board. The LED is connected to HPS_GPIO53, KEY is connected to HPS_GPIO54, which are controlled by the GPIO1 controller, which also controls HPS_GPIO29 ~ HPS_GPIO57.

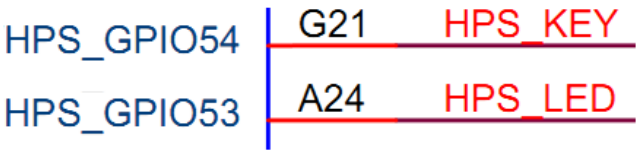


Figure 6-4 LED and KEY Pin Assignment

Figure 6-5 shows the **gpio_swporta_ddr** register of the GPIO1 controller. The bit-0 controls the pin direction of HPS_GPIO29. The bit-24 controls the pin direction of HPS_GPIO53, which connects to the HPS_LED, the bits-25 controls the pin direction of HPS_GPIO54 which connects to the HPS_KEY, and so on. In summary, the pin direction of HPS_LED, HPS_KEY are controlled by the bit-24, bit-25 in the **gpio_swporta_ddr** register of the GPIO1 controller, respectively. Similarly, the output status of HPS_LED is controlled by the bit-24 in the **gpio_swporta_dr** register of the GPIO1 controller. The status of KEY can be queried by reading the value of the bit-24 in the **gpio_ext_porta** register of the GPIO1 controller.

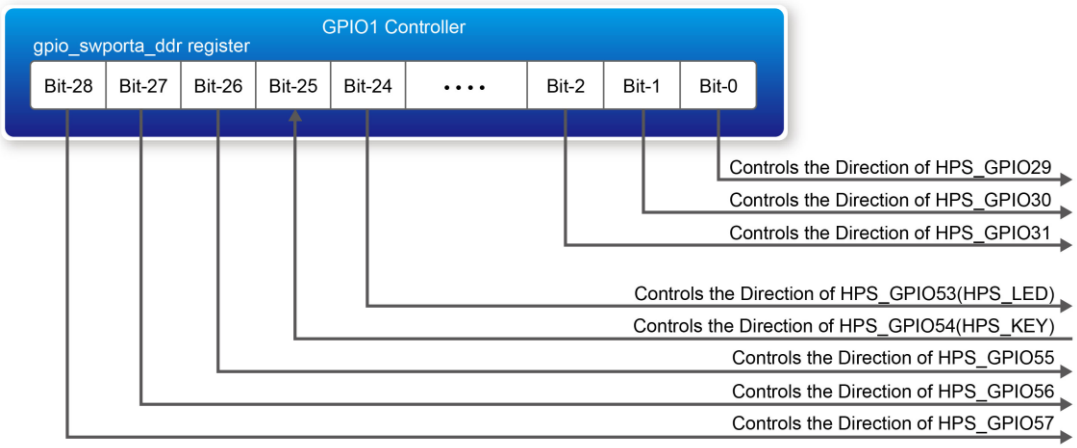


Figure 6-5 gpio_swporta_ddr Register in the GPIO1

In this demo code, the following mask is defined to control LED and KEY direction and LED's output value.

```
#define USER_IO_DIR          (0x01000000)

#define BIT_LED              (0x01000000)

#define BUTTON_MASK          (0x02000000)
```

The following statement can be used to configure the LED associated pins as output pins.

```
alt_setbits_word( ( virtual_base +
( ( uint32_t )( ALT_GPIO1_SWPORTA_DDR_ADDR ) &
( uint32_t )( HW_REGS_MASK ) ) ), USER_IO_DIR );
```

The following statement can be used to turn on the LED.

```
alt_setbits_word( ( virtual_base +
( ( uint32_t )( ALT_GPIO1_SWPORTA_DR_ADDR ) &
( uint32_t )( HW_REGS_MASK ) ) ), BIT_LED );
```

The following statement can be used to read the content of **gpio_ext_porta** register. The bit mask is used to check the status of the key.

```
alt_read_word( ( virtual_base +
( ( uint32_t )(    ALT_GPIO1_EXT_PORTA_ADDR ) &
( uint32_t )( HW_REGS_MASK ) ) ) );
```

.

■ Demonstration Source Code

- Build tool: Altera SoC EDS V13.1
- Project directory: \Demonstration\SoC\hps_gpio
- Binary file: hps_gpio
- Build command: make ('make clean' to remove all temporal files)
- Execute command: ./hps_gpio

■ Demonstration Setup

- Connect the USB cable to the USB-to-UART connector (J4) on the DE1_SoC board and host PC.
- Make sure the executable file "**hps_gpio**" is copied into the SD card under the "**/home/root**" folder in Linux.
- Insert the booting micro SD card into the DE1_SoC board.
- Power on the DE1_SoC board.
- Launch PuTTY to connect to the UART port of DE1_SoC board and type "**root**" to login Altera Yocto Linux.
- In the UART terminal of PuTTY, execute "**./hps_gpio**" to start the program.

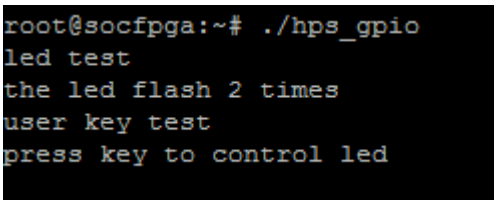


Figure 6-6 Putty window

- HPS_LED will flashing 2 times first and the user can control the user LED with user Button.
- Press HPS_KEY to light up HPS_LED ,Press "CTRL + C" to terminate the application.

6.3 I2C Interfaced G-sensor

This demonstration shows how to control the G-sensor by accessing its registers through the built-in I2C kernel driver in Altera Soc Yocto Powered Embedded Linux.

■ Function Block Diagram

Figure 6-7 shows the function block diagram of this demonstration. The G-sensor on the DE1_SoC board is connected to the **I2C0** controller in HPS. The G-Sensor I2C 7-bit device address is 0x53. The system I2C bus driver is used to access the register files in the G-sensor. The G-sensor interrupt signal is connected to the PIO controller. In this demonstration, we use polling method to read the register data, so the interrupt method is not introduced here.

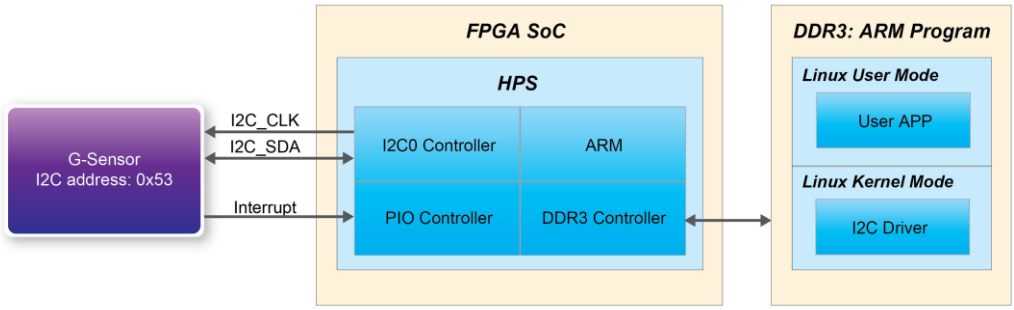


Figure 6-7 Block Diagram of the G-sensor Demonstration

■ I2C Driver

Here is the list of procedures in order to read a register value from G-sensor register files by using the existing I2C bus driver in the system:

1. Open I2C bus driver "/dev/i2c-0": file = open("/dev/i2c-0", O_RDWR);
2. Specify G-sensor's I2C address 0x53: ioctl(file, I2C_SLAVE, 0x53);
3. Specify desired register index in g-sensor: write(file, &Addr8, sizeof(unsigned char));
4. Read one-byte register value: read(file, &Data8, sizeof(unsigned char));

Because the G-sensor I2C bus is connected to the I2C0 controller, as shown in the **Figure 6-8**, the given driver name is '/dev/i2c-0'.

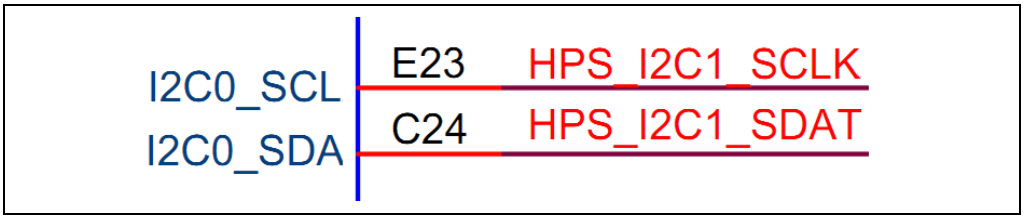


Figure 6-8 Schematic of I2C

To write a value into a register, developer can change step 4 to:

write(file, &Data8, sizeof(unsigned char));

To read multiple byte values, developer can change step 4 to:

read(file, &szData8, sizeof(szData8)); // where szData is an array of bytes

To write multiple byte values, developer can change step 4 to:

write(file, &szData8, sizeof(szData8)); // where szData is an array of bytes

■ G-sensor Control

The ADI ADXL345 provides I2C and SPI interfaces. I2C interface is used by setting the CS pin to high on this DE1_SoC board.

The ADI ADXL345 G-sensor provides user-selectable resolution up to 13-bit ± 16g. The resolution can be configured through the DATA_FORAMT(0x31) register. In the demonstration, we configure the data format as:

- Full resolution mode
- ± 16g range mode
- Left-justified mode

The X/Y/Z data value can be derived from the DATAX0(0x32), DATAX1(0x33), DATAY0(0x34), DATAY1(0x35), DATAZ0(0x36), and DATAX1(0x37) registers. The DATAX0 represents the least significant byte, and DATAX1 represents the most significant byte. It is recommended to perform multiple-byte read of all registers to prevent change in data between reads of sequential registers. Developer can use the following statement to read 6 bytes of X, Y, or Z value.

read(file, szData8, sizeof(szData8)); // where szData is an array of six-bytes

■ Demonstration Source Code

- Build tool: Altera SoC EDS v13.1
- Project directory: \Demonstration\SoC\hps_gsensor
- Binary file: gsensor
- Build command: make ('make clean' to remove all temporal files)
- Execute command: ./gsensor [loop count]

■ Demonstration Setup

- Connect the USB cable to the USB-to-UART connector (J4) on the DE1-SoC board and host PC.
- Make sure the executable file "**gsensor**" is copied into the SD card under the "**/home/root**" folder in Linux.
- Insert the booting micro sdcard into the DE1-SoC board.

- Power on the DE1-SoC board.
- Launch PuTTY to connect to the UART port of DE1-SoC borad and type "**root**" to login Yocto Linux.
- In the UART terminal of PuTTY,, execute "**./gsensor**" to start the gsensor polling.
- The demo program will show the X, Y, and Z values in the Putty, as shown in Figure 6-9. Press "CTRL + C" to terminate the program.

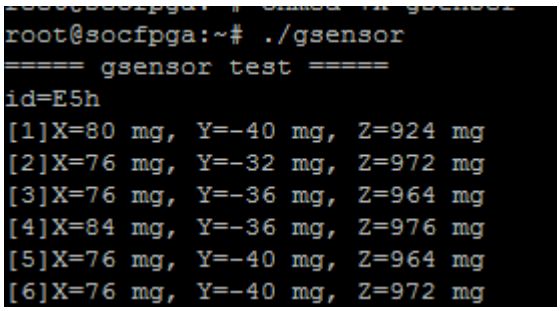


Figure 6-9 Terminal output of the G-sensor Demonstration

6.4 I2C MUX Test

This demonstration shows how to switch the I2C multiplexer so that HPS can access the I2C bus originally owned by FPGA.

■ Function Block Diagram

Figure 6-10 shows the function block diagram of this demonstration. The I2C bus from both FPGA and HPS are connected to an I2C multiplexer and I2C multiplexer is controlled by HPS_I2C_CONTROL which connected to **GPIO1** controller in HPS. The HPS I2C is connected to the **I2C0** controller in HPS (Gsensor is also connected to I2C0 controller, See Figure 6-9).

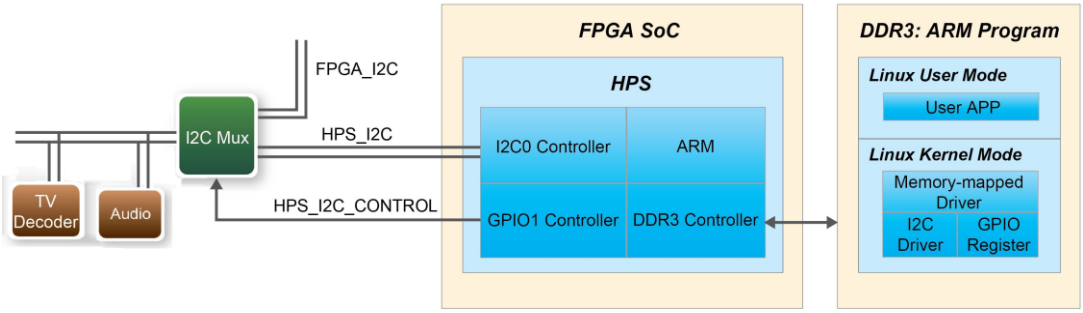


Figure 6-10 Block Diagram of the I2C MUX Test Demonstration

- **HPS_I2C_CONTROL Control**

HPS_I2C_CONTROL is connected to HPS_GPIO48, bit 19 of **GPIO1** controller. HPS will own I2C bus and then can access Audio CODEC and TV Decoder when the HPS_I2C_CONTROL signal is set to high.

In this demo code, the following mask is defined to control HPS_I2C_CONTROL direction and their output value.

```
#define HPS_I2C_CONTROL      ( 0x00080000 )
```

The following statement can be used to configure the HPS_I2C_CONTROL associated pins as output pin.

```
alt_setbits_word( ( virtual_base +
( ( uint32_t )( ALT_GPIO1_SWPORTA_DDR_ADDR ) &
( uint32_t )( HW_REGS_MASK ) ) ), HPS_I2C_CONTROL );
```

The following statement can be used to set HPS_I2C_CONTROL high.

```
      alt_setbits_word( ( virtual_base +
( ( uint32_t )( ALT_GPIO1_SWPORTA_DR_ADDR ) &
( uint32_t )( HW_REGS_MASK ) ) ), HPS_I2C_CONTROL );
```

The following statement can be used to set HPS_I2C_CONTROL low.

```
      alt_clrbits_word( ( virtual_base +
( ( uint32_t )( ALT_GPIO1_SWPORTA_DR_ADDR ) &
( uint32_t )( HW_REGS_MASK ) ) ), HPS_I2C_CONTROL );
```

- **I2C Driver**

Here is the list of procedures in order to read register value from TV Decoder by using the existing I2C bus driver in the system:

- Set HPS_I2C_CONTROL high so that HPS can access I2C bus.
- Open I2C bus driver "/dev/i2c-0": file = open("/dev/i2c-0", O_RDWR);
- Specify ADV7180 's I2C address 0x20: ioctl(file, I2C_SLAVE, 0x20);
- Read or write registers;
- Set HPS_I2C_CONTROL low to release I2C bus.

- **Demonstration Source Code**

- Build tool: Altera SoC EDS v13.1
- Project directory: \Demonstration\SoC\ hps_i2c_switch
- Binary file: i2c_switch
- Build command: make ('make clean' to remove all temporal files)
- Execute command: ./ i2c_switch

■ Demonstration Setup

- Connect the USB cable to the USB-to-UART connector (J4) on the DE1_SoC board and host PC.
- Make sure the executable file " **i2c_switch** " is copied into the SD card under the "**/home/root**" folder in Linux.
- Insert the booting micro sdcard into the DE1_SoC board.
- Power on the DE1-SoC board.
- Launch PuTTY to connect to the UART port of DE1_SoC borad and type "**root**" to login Yocto Linux.
- In the UART terminal of PuTTY,, execute "**./ i2c_switch** " to start the I2C MUX test.
- The demo program will show the result in the Putty, as shown in **Figure 6-11**.

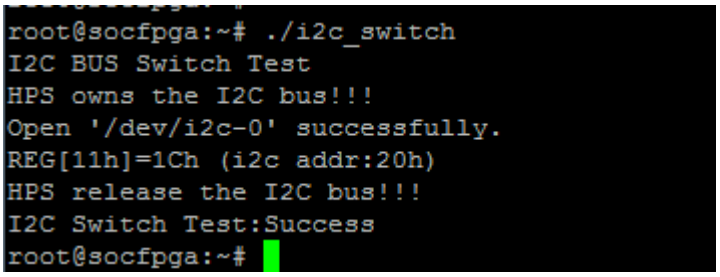


Figure 6-11 Terminal output of the I2C MUX Test Demonstration

- Press "CTRL + C" to terminate the program.

Chapter 7

Examples for using both HPS SoC and FGPA

Although the HPS and the FPGA can operate independently, they are tightly coupled via a high-bandwidth system interconnect built from high-performance ARM AMBA® AXITM bus bridges. Both FPGA fabric and HPS can access to each other via these interconnect bridges. This chapter provides demonstrations for how to using these bridges that can achieve superior performance and lower latency when compared to solutions containing a separate FPGA and discrete processor.

7.1 HPS Control LED and HEX

This demonstration presents using HPS to configure FPGA through FPGA manager in HPS and control the LED and HEX on the FPGA part through Lightweight HPS-to-FPGA Bridge.

- **A brief view on FPGA manager**

The FPGA manager in HPS can configure the FPGA fabric from the HPS, monitor the state of the FPGA, and drive or sample signals to or from the FPGA fabric. We provide application software to configure FPGA through the FPGA manager. The FPGA configure data is stored in the file with .rbf extension. Before executing the application software on HPS, users should check the MSEL[4:0] is set as 01010 or 01110.

- **Function Block Diagram**

Figure 7-1 shows the diagram of this demonstration. The HPS use Lightweight HPS-to-FPGA AXI Bridge to communicate with FPGA. The HPS translate data to the FPGA through the lwaxi bridge.

The hardware in FPGA part is built in Qsys. The data translate through Lightweight HPS-to-FPGA Bridge is converted into Avalon-MM master interface. So the IP PIO controller and HEX Controller works as the Avalon-MM slave in the system. They control the pins related to the LED and HEX to change the LED and HEX's state. This is similar to the system using NIOS II processor to control LED and HEX.

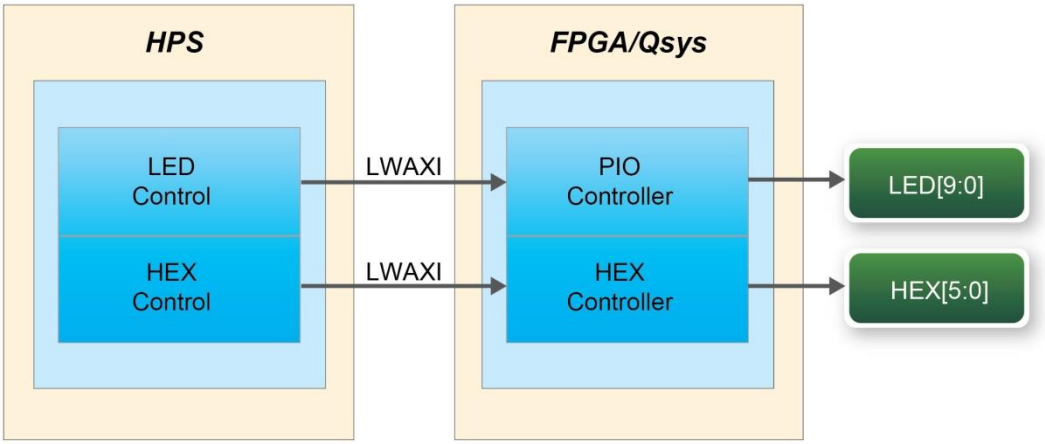


Figure 7-1 HPS Control FPGA LED and HEX

■ LED and HEX control

The Lightweight HPS-to-FPGA Bridge is a peripheral of the HPS. The software running on linux operation system can't access the physical address of the HPS peripheral. You must map the physical address to the user space at first then you can access to the peripheral or you can write a device driver module and add it to the kernel. We only show the first method to the users in this demonstration. We actually map in the entire CSR span of the HPS since we want to access various registers within that span. If the users want to access any other peripherals whose physical address is in this span, they can reuse the mapping function and the macro we defined below.

```
#define HW_REGS_BASE ( ALT_STM_OFST )
#define HW_REGS_SPAN ( 0x04000000 )
#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )
```

The lwaxi bridge start address after being mapped can be get using the ALT_LWFPGASLVS_OFST which is defined in altera_hps hardware library. Then the slave IP connected to the lwaxi bridge can be accessed through the base address and the register offset in these IPs. For instance, the base address of the PIO slave IP in this system is 0x0001_0040 and the direction control register offset is 0x01, the data register offset is 0x00. The following statement can be used to get the base address of PIO slave IP.

```
h2p_lw_led_addr=virtual_base+( ( unsigned long    )( ALT_LWFPGASLVS_OFST
+ LED_PIO_BASE ) & ( unsigned long)( HW_REGS_MASK ) );
```

In this demonstration, we just need to set the PIO's direction as output which is the default direction of the PIO IP, so we can skip this step. The following statement is used to set the output state of the PIO.

```
alt_write_word(h2p_lw_led_addr, Mask );
```

The Mask in this statement decides which bit in the data register of the PIO IP is high or low. The bits in data register decide the output state of the pins connected to the LEDs. The program for the HEX controlling is similar to the LED.

Since the linux operate system support mult-thread software. The software for this system creates two threads one for controlling the LED and the other for controlling the HEX. We can use the system call pthread_create to complete the job. The function is called in the main function and a sub-thread is created. The program running in the sub-thread is to control the led flashing in a loop. And the main-thread in the main function is to control the digital shown on the HEX changing in a loop. The LED and HEX's state changing at the same time when the FPGA is configured and the software is running on HPS.

■ **Demonstration Source Code**

- Build tool: Altera SoC EDS V13.1
- Project directory: \Demonstration\ SoC_FPGA\HPS_LED_HEX
- Quick file directory:\ Demonstration\ SoC_FPGA\HPS_LED_HEX\ quickfile
- FPGA Configure File : soc_system_dc.rbf
- Binary file: HPS_LED_HEX and hps_config_fpga
- Build app command: make ('make clean' to remove all temporal files)
- Execute app command:./hps_config_fpga soc_system_dc.rbf and./HPS_LED_HEX

Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Make sure the MSEL[4:0] on DE1-SoC board is set to 01010 or 01110.
- Connect the USB blaster cable to the USB blaster connector (J13) on the DE1-SoC board and host PC install USB Blaster driver II if necessary.
- Connect the USB cable to the USB-to-UART connector (J4) on the DE1-SoC board and host PC.
- Make sure the executable file "hps_config_fpga" and "HPS_LED_HEX" and the FPGA

configure file "soc_system_dc.rbf" are copied into the SD card under the "**/home/root**" folder in Linux.

- Insert the booting micro SD card into the DE1-SoC board. (For how to build a booting micro SD card image, Please refer to the chapter 5 "Running Linux on the DE1-SoC board" on *DE1-SoC_Getting_Started_Guide.pdf*)
- Power on the DE1-SoC board.
- Launch PuTTY to connect to the UART port of DE1-SoC board and type "**root**" to login Altera Yocto Linux.
- In the UART terminal of PuTTY, execute "./hps_config_fpga soc_system_dc.rbf " to configure the FPGA through the FPGA manager. After configuring done, the message shown in the Putty as shown in **Figure 7-2.**

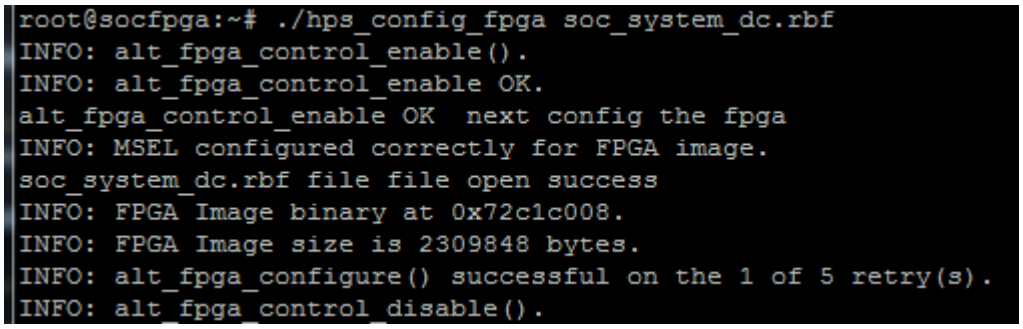


Figure 7-2 running app configure FPGA

- In the UART terminal of PuTTY, execute "**./**HPS_LED_HEX " to start the program.
- The putty will show the message as shown in **Figure 7-3**, and the LED[9:0] will flash , the number on the HEX[5:0] will change at the same time. Press "CTRL + C" to terminate the program.

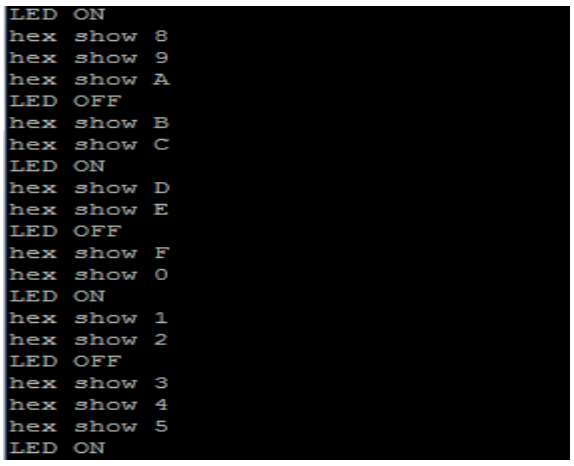


Figure 7-3 Running result in putty

Chapter 8

Steps of Programming the Quad Serial Configuration Device

This chapter describes how to program the quad serial configuration device with Serial Flash Loader (SFL) function via the JTAG interface. User can program quad serial configuration devices with a JTAG indirect configuration (.jic) file. To generate JIC programming files with the Quartus II software, users need to generate a user-specified SRAM object file (.sof), which is the input file first. Next, users need to convert the SOF to a JIC file. To convert a SOF to a JIC file in Quartus II software, follow these steps:

8.1 Before you Begin

To use the Quad serial flash as a FPGA configuration device, please make sure the FPGA should be set in Asx4 mode. (i. e. , let MSEL[4..0] to be set as “10010”)

8.2 Convert. SOF File to .JIC file

Choose **Convert Programming Files** on Quartus window (File menu). See **Figure 8-1**

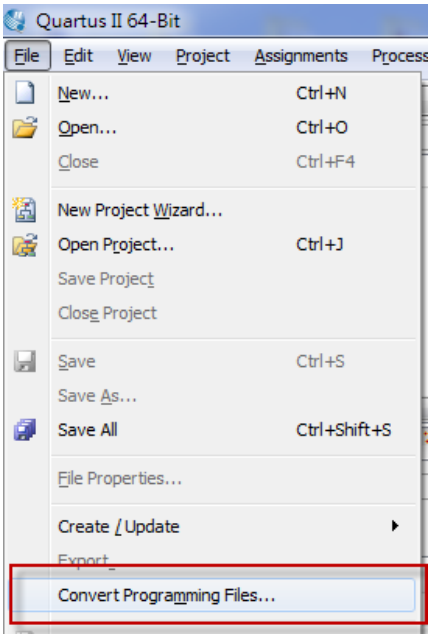


Figure 8-1 File menu of Quartus

In the Convert Programming Files dialog box, scroll to the JTAG Indirect Configuration File (.jic) from the Programming file type field.

In the Configuration device field, choose EPCQ256.

In the **Mode** field, choose **Active Serial X4**.

In the **File name** field, browse to the target directory and specify an output file name.

Highlight the SOF data in the Input files to convert section. See **Figure 8-2**.

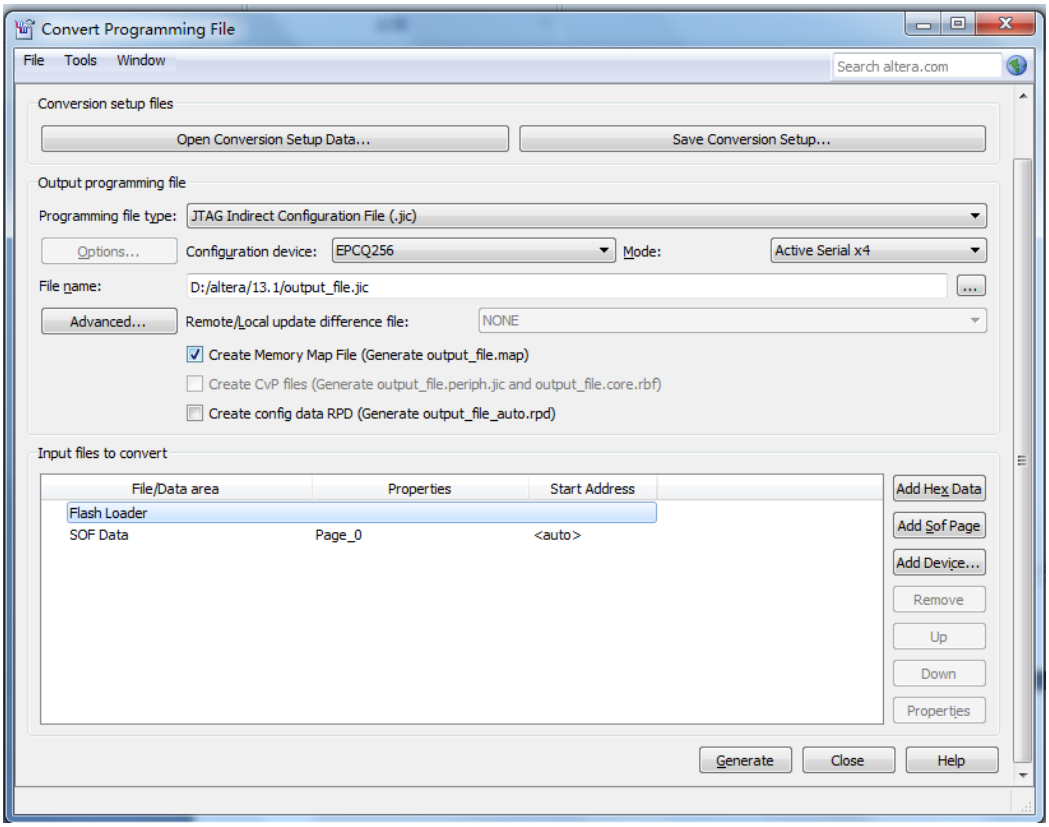


Figure 8-2 Convert Programming Files Dialog Box

Click Add File.

Select the SOF that you want to convert to a JIC file.

Click **Open**.

Highlight the Flash Loader and click Add Device. See **Figure 8-3.**

Click **OK**. The Select Devices page displays.

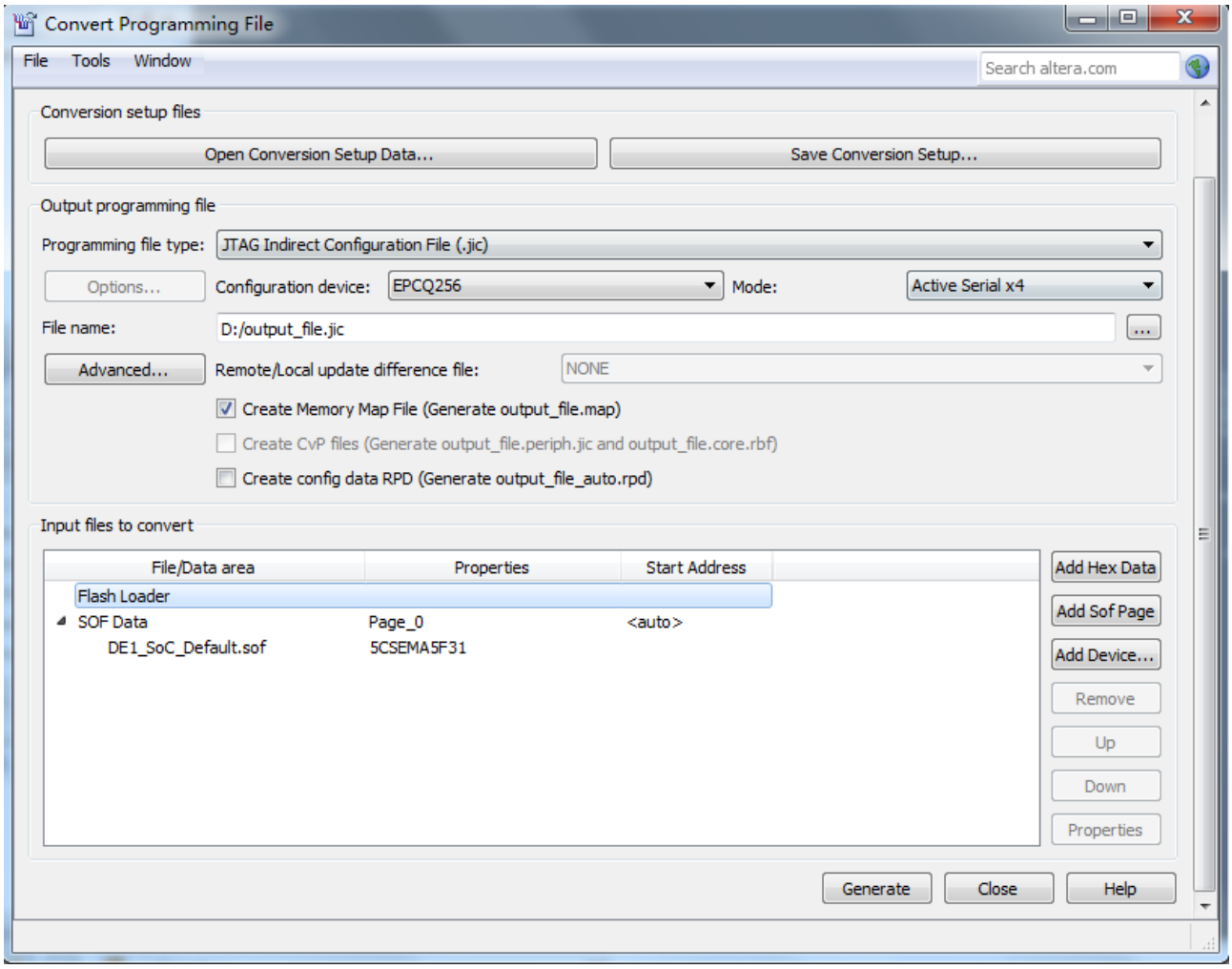


Figure 8-3 Highlight Flash Loader

Select the targeted FPGA that you are using to program the serial configuration device. See **Figure 8-4**.

Click OK. The Convert Programming Files page displays. See **Figure 8-5**.

Click Generate.

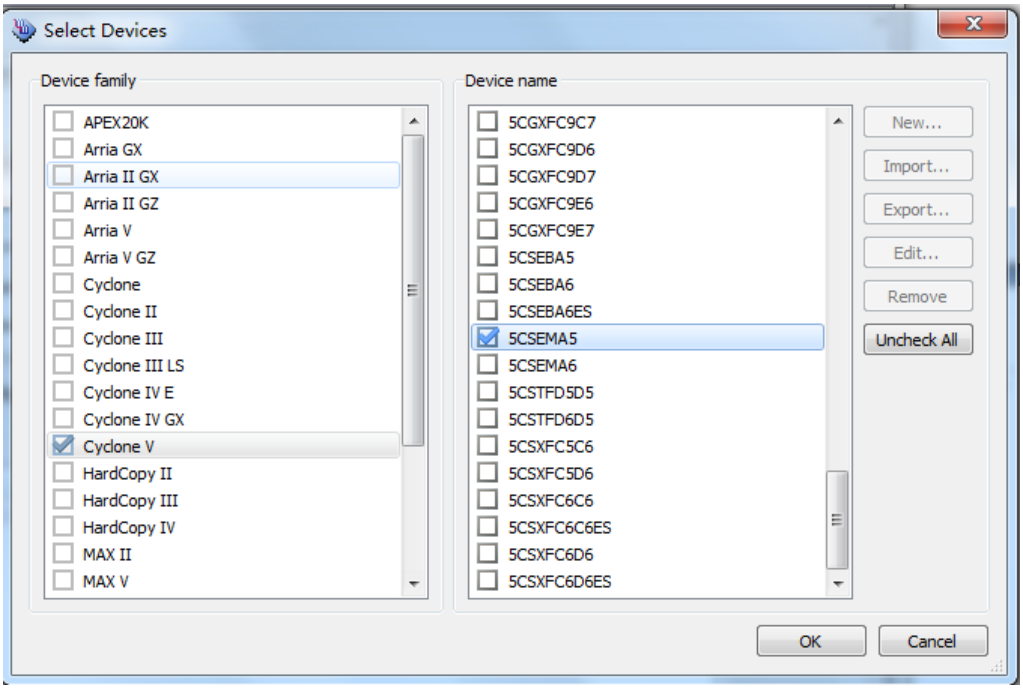


Figure 8-4 Select Devices Page

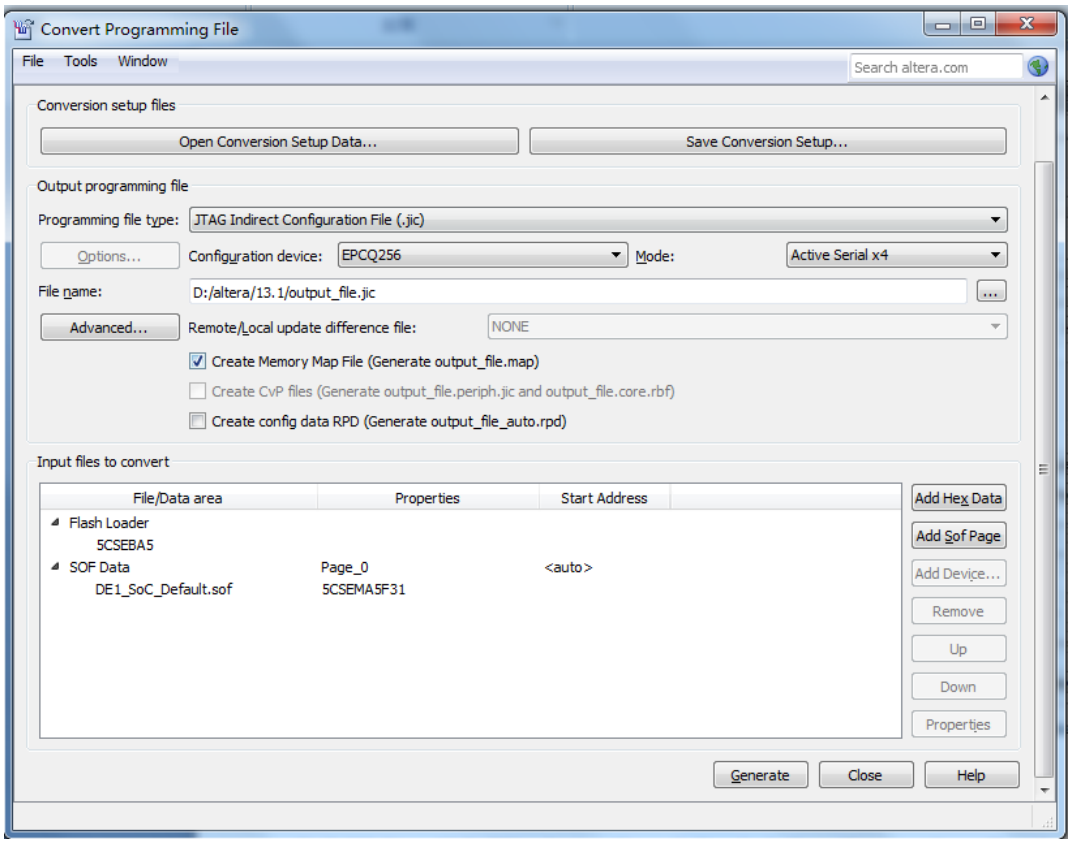


Figure 8-5 Convert Programming Files Page

8.3 Write JIC File into Quad Serial Configuration Device

To program the serial configuration device with the JIC file that you just created, add the file to the Quartus II Programmer window and follow the steps:

When the SOF-to-JIC file conversion is complete, add the JIC file to the Quartus II Programmer window:
Choose **Programmer** (Tools menu), and the **Chain.cdf** window appears.

Click **Auto Detect**, and then select correct device, both FPGA device and HPS will detected. (See **Figure 8-6)**
Double click the green rectangle region as shown in **Figure 8-6**, the **Select New Programming File** page will appear, and then select the correct JIC file.

Program the serial configuration device by clicking the corresponding **Program/Configure** box, a factory default SFL image will be loaded (See **Figure 8-7**).

Click **Start** to program serial configuration device.

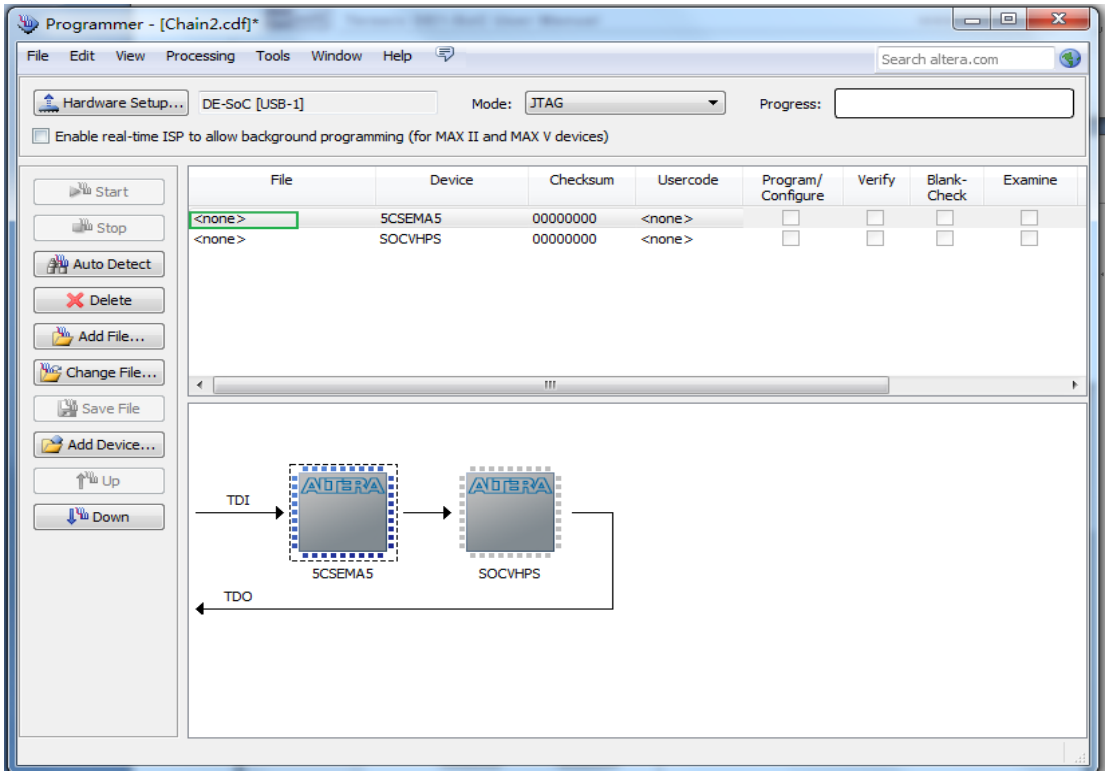


Figure 8-6 Quartus II programmer window with two detected devices

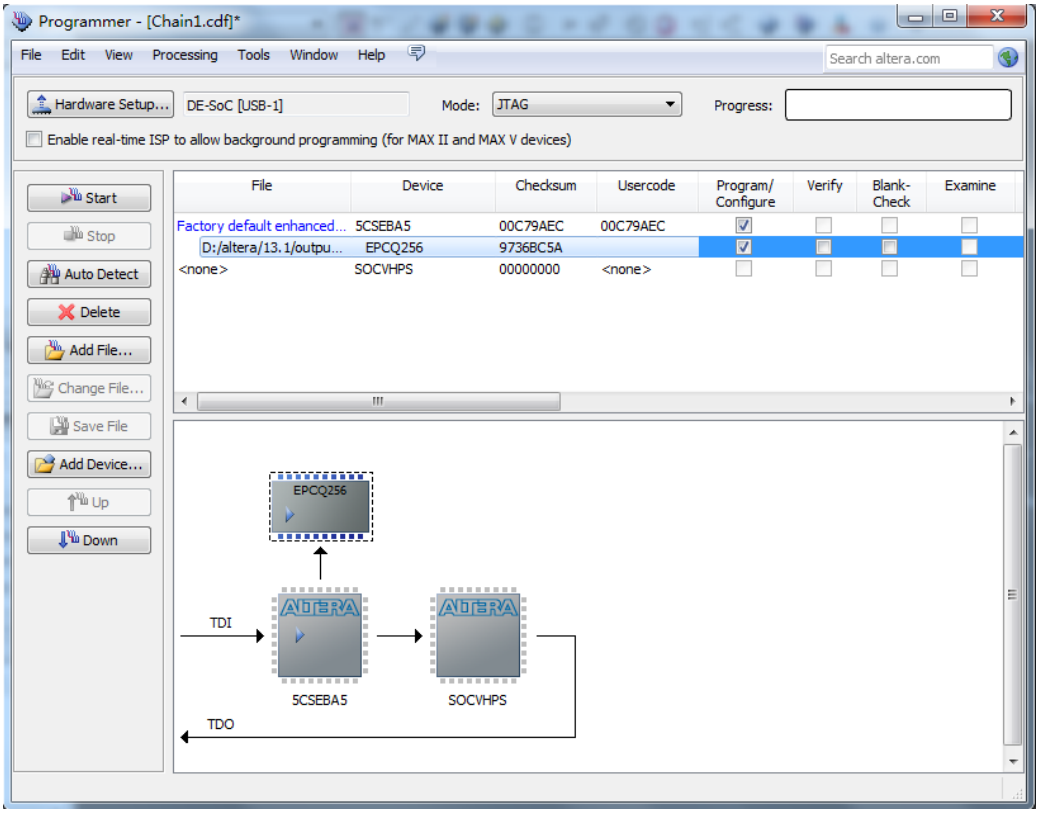


Figure 8-7 Quartus II programmer window with one JIC file

8.4 Erase the Quad Serial Configuration Device

To erase the existed file in the serial configuration device, follow the steps listed below:

Choose **Programmer** (Tools menu), and the **Chain.cdf** window appears.

Click **Auto Detect**, and then select correct device, both FPGA device and HPS will detected. (See **Figure 8-6)**

Double click the green rectangle region as shown in **Figure 8-6**, the **Select New Programming File** page will appear, and then select the correct JIC file.

Erase the serial configuration device by clicking the corresponding **Erase** box, a Factory default SFL image will be load (See **Figure 8-8**).

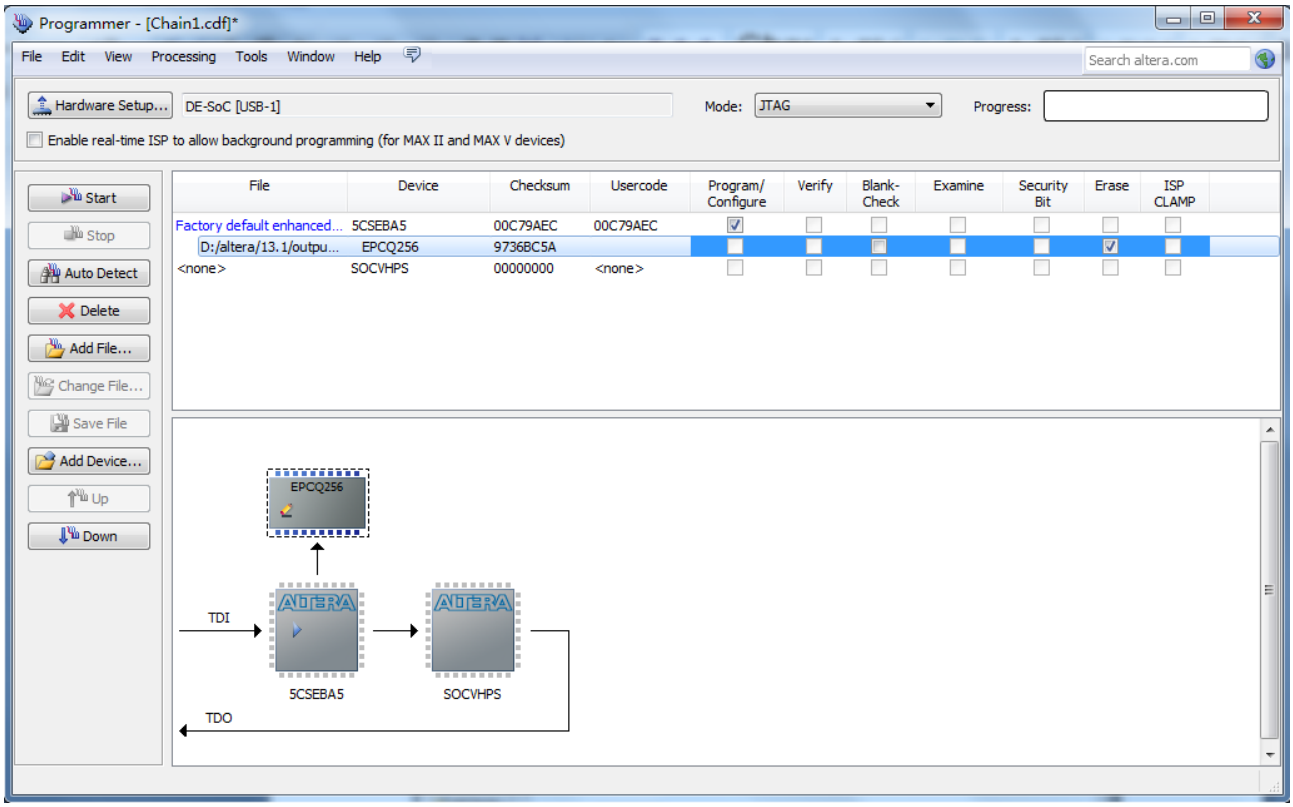


Figure 8-8 Erasing setting in Quartus II programmer window

Click **Start** to erase the serial configuration device.

Chapter 9

Appendix

9.1 Revision History

Version	*Change Log*
V0.1	Initial Version (Preliminary)
V0.2	Add CH5 and CH6
V0.3	Modify CH3
V0.4	Add CH6 HPS
V0.5	Modify CH3
V1.0	Modify CH8

Copyright © 2014 Terasic Technologies. All rights reserved.